Análisis de tecnología para la implementación de arquitecturas de procesamiento de Grandes Datos

Daniel Molina Ruiz
Universidad Carlos III de Madrid
Septiembre 2017

Agradecimientos

Este Proyecto Fin de Carrera supone el cierre de una etapa de mi vida que aunque parecía haber quedado atrás, nunca lo estuvo del todo: La Espada de Damocles que siempre me ha acompañado y que por fin ahora puedo descolgar de su crin y enfundar sin peligro.

Quiero agradecer a mi tutor José María Álvarez Rodríguez el haberme brindado la oportunidad de realizar este proyecto. Su guía en la andadura de su elaboración, reorientándome cuando me desviaba, y su diligente ayuda ante los obstáculos que encontré, sin duda han sido claves para poder llevarlo a buen término.

Gracias a mis compañeros, con quienes tanto tiempo pasé en el campus de Leganés- que vimos crecer a nuestro alrededor con el devenir de los años- por todas vivencias compartidas y el apoyo que siempre me brindaron. En especial, gracias a Luis, que no sólo fue un magnífico compañero, sino que se convirtió en un gran amigo.

Mi sincera e infinita gratitud a mi familia que, a pesar de mi demora, nunca perdió la esperanza ni dejó de apoyarme en la consecución de este trabajo. Gracias a mis padres, por haberme conducido hasta aquí y permitir convertirme en lo que hoy soy. Un cariñoso recuerdo para mis abuelos, quienes seguro estarían orgullosos de este logro y con quienes lamento no poder compartirlo.

Y finalmente, gracias a mi amiga, mi compañera, mi confidente, Vanesa. Gracias por tu apoyo y por tu empuje, tus ánimos y tus collejas. Por los sacrificios y la comprensión. Por fin lo conseguimos.

Gracias a todos, por todo.

Resumen

En la actualidad, el desarrollo del ancho de banda en los sistemas de comunicaciones y el aumento de la capacidad de los dispositivos de almacenamiento, ha dado lugar a que los volúmenes de información disponible para ser tratada hayan crecido enormemente, dando lugar a los Grandes Datos o Big Data. Pese a que los procesadores de ordenadores personales y servidores también han visto incrementada su potencia sustancialmente, se hace necesario un nuevo modo de tratar con toda esta información.

El objetivo de este Proyecto Fin de Carrera es evaluar las capacidades de una herramienta para el tratamiento de Grandes Datos para su implementación en una red de comunicaciones existente teniendo en cuenta sus características, necesidades y restricciones particulares.

El tamaño de la red de comunicaciones y la diversidad de fuentes de información presentes en la misma hacen imposible la implementación completa de la solución necesaria para resolver el problema. Por este motivo se optó por implementar una arquitectura genérica más reducida y alimentarla con fuentes de datos específicas de esta red de comunicaciones concreta en lugar de otras estándar. Con este montaje se realizaron pruebas de evaluación y validación de la herramienta que permitieron concluir su idoneidad para el caso objeto de estudio.

Aunque se deja para trabajo futuro la implementación práctica de la herramienta de Grandes Datos y su integración en la red de comunicaciones, se ha realizado un análisis de las características de todas las fuentes de información presentes en la misma, así como naturaleza de la información proveniente de cada una de ellas. También se propone una arquitectura completa capaz de prestar el servicio.

Abstract

Nowadays, bandwidth increase in communications systems and capacity enhancement in storage devices has resulted in a huge growth in volume of information available for processing, which is known as Big Data. Even though processors in personal computers and enterprise servers have seen their performance substantially improve as well, a new way to deal with all this information has become necessary.

The goal for this Final Degree Project is to evaluate capacities for a Big Data management tool, in order to implement it on an existing communications network, taking into consideration its particular characteristics, needs and restrictions.

Size of the communications network and variety of all its sources of information make it impossible to implement the full system that would be necessary to tackle the problem. For this reason, it was decided to implement a generic setup, smaller and feed it with sources of data specific to this communications network, instead of any other standard ones. Evaluation and validation test were undergone on this setup, and let us conclude it was suitable for the specific scenario.

Even though practical implementation of this Big Data management tool and its integration into the communications network has been postponed for future work, all sources of information present on it have been analyzed, and so has the nature of information generated by each of them. A full architecture, capable of delivering the service has also been proposed.

Índice

Introducción

El objetivo de este estudio es el análisis de la oferta actual de soluciones para el tratamiento de Big Data, y la elección de una adecuada para su aplicación a los registros generados por una red concreta de comunicaciones, teniendo en cuenta las características de los mismos.

Estructura de la memoria

Se comenzará con una breve explicación de lo que se conoce como Big Data y el modo en que históricamente se comenzó a dar solución a su problemática. A continuación se describirán soluciones de libre distribución que enfocan el problema desde diferentes aproximaciones, para entonces presentar algunos productos completos enfocados específicamente al tratamiento de grandes cantidades de logs.

Con estas bases, pasarán a describirse los requisitos de la plataforma de gestión de eventos para el caso concreto de la red de comunicaciones objeto de estudio, siguiendo con una propuesta de diseño, y la evaluación práctica de ciertas funcionalidades de la misma.

Se finalizará con una exposición de las conclusiones extraídas a partir de los resultados obtenidos, y la propuesta de los siguientes pasos a tomar de cara a la implantación de la solución expuesta.

Objetivos

Ante la amplia oferta de productos existentes para el tratamiento de Grandes Datos, en este trabajo se pretende revisar en primer lugar los más populares de entre aquellos enfocados a la gestión de registros, y escoger el que reúna las características más adecuadas a las necesidades de la red de comunicaciones (expuestas en el apartado Análisis).

El segundo paso consistirá en el diseño de la arquitectura que deberá implementarse con el producto escogido, comprobando que sus componentes y funcionalidades cumplen, efectivamente, con los requisitos recogidos.

Para concluir, se evaluará el funcionamiento del producto con datos reales tomados de la red de comunicaciones, verificando su comportamiento ante situaciones de caída o su rendimiento en función de varios parámetros, tales como volumen de información, definición de nuevos índices o calidad de las consultas.

Estado del arte

Big Data

El término Big Data se emplea para referirse a un conjunto de datos que resulta demasiado grande o complejo como para ser manejado con herramientas tradicionales. Los retos que presenta incluyen la captura, el análisis, la conservación[1] (*data curation*), el almacenamiento, la consulta, la búsqueda, la compartición, la transferencia, la visualización, la actualización o la privacidad de la información.

En los últimos años el crecimiento de la información disponible ha venido dado en gran medida por la proliferación de nuevos métodos para captación de la misma: dispositivos móviles, lectores de identificación por radiofrecuencia (RFID), redes de sensores inalámbricos, etc. han venido a sumarse a los tradicionales logs generados por dispositivos y software clásicos.
Blogs, webs y audio o video en streaming también suponen una importante fuente de información. De este modo, en consecuencia, los datos disponibles pueden ser estructurados, semi-estructurados o sin estructurar, un hecho que caracteriza al Big Data.

Además, la evolución de la tecnología también ha posibilitado que la misma observación pueda ser registrada de forma mucho más precisa, aumentando la información disponible para la misma.
Por ejemplo, el Telescopio SDSS recogió en sus primeras semanas de funcionamiento, en el año 2000, más datos que en toda la historia previa de la astronomía[i]. Una década más tarde sus archivos contaban con 140 TB de información. Se espera que su sucesor, el LSST, actualmente en construcción, recabe toda esa información aproximadamente cada semana. Pero también las capacidades de procesamiento han aumentado de forma similar: Cuando el genoma humano fue secuenciado por primera vez en 2003 supuso un trabajo de 10 años, mientras que esta cifra se ha reducido a horas en la actualidad[ii].

Un aspecto importante que también debe tomarse en cuenta hace referencia a la "relatividad" del término "Big Data", en función de la capacidad de los usuarios y herramientas que se enfrentan al conjunto de datos. Ciertas organizaciones pueden comenzar a tener problemas a la hora de manejar conjuntos de cientos de gigabytes de información, mientras que los gigantes como Amazon, Facebook o Walmart manejan incluso cientos de terabytes sin mayor problema.

Existen diferentes modos de caracterizar el Big Data, siendo el Multi-V uno de los más populares[iii].
La diversidad de las fuentes y tipos de datos conforman la **V**ariedad (estructurado, sin estructurar,…), mientras que el ritmo al que se generan determina la **V**elocidad y el **V**olumen describe la cantidad de datos (estático, considerado en un momento dado). Igualmente importantes son la **V**eracidad, que se refiere a la fiabilidad que tienen los datos (inconsistencias, vacíos, incertidumbre), y el **V**alor de los mismos, evaluarlos,

[1] Entendida como la gestión de los datos a lo largo de su ciclo de vida, desde su creación y almacenamiento inicial, pasando por su almacenamiento indefinido (manteniéndolos accesibles sin importar que puedan cambiar de ubicación), hasta que se hacen obsoletos y son eliminados.

pues disponer de ellos no significa necesariamente que merezca la pena procesarlos o explotarlos.
No tan habituales pero también utilizados, serían la **V**irología (por no utilizar la adaptación directa de Virality, Viralidad) que describe lo rápido que se propagan los datos por las redes, y la **V**iscosidad, que mide la resistencia se presenta para navegar por el conjunto de datos (por su variedad de fuentes o por la complejidad de la generación y procesamiento de los datos).

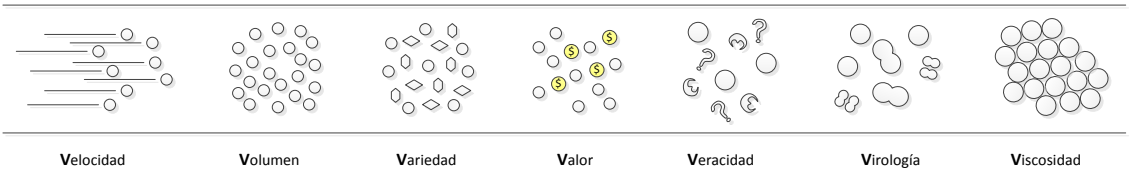


Figura 1. Modelo Multi-V

El modo de enfrentarse al Big Data varía en función de las necesidades y posibilidades de que se disponga, así como, entre otros factores, de la ubicación de los datos. De este modo, las opciones que se presentan son el uso de una infraestructura dedicada o el uso en la nube.
Esta última opción viene tomando fuerza en los últimos tiempos[iv] impulsada en gran medida por el enorme crecimiento de datos disponibles en la nube (se estimaba que en 2016, la mitad de los datos mundiales se encontrarían en ella), así como el desarrollo de la misma como plataforma de almacenamiento y procesamiento.
Todo ello la convierte en una solución muy atractiva, no sólo desde el punto de vista económico, sino también por la liberación para el usuario de los aspectos inherentes al mantenimiento de los servicios que provee. Por el contrario, la propia naturaleza distribuida de la nube también se convierte en su mayor inconveniente, y en la actualidad mucho esfuerzo se está dedicando optimizar el rendimiento[2] de las plataformas Big Data en ella. Así mismo, en la nube la seguridad y privacidad corren el riesgo de verse comprometidas, motivo por el cual se están proponiendo nuevas soluciones para poder enfrentarse a este desafío.

MapReduce

MapReduce surge en 2004, desde Google, como un modelo de programación para el procesado de grandes conjuntos de datos. Las operaciones a realizar no eran computacionalmente exigentes, pues consistían en indexaciones inversas, resúmenes del número de páginas visitadas por un host o representaciones de la estructura de documentos web.
El desafío cada vez mayor, venía dado por la ingente cantidad de datos a procesar y el reducido tiempo para llevarlo a cabo, que requería del uso de miles de máquinas entre las que repartir el trabajo. La distribución de los datos, gestión de los trabajos en

[2] De cara al usuario, la velocidad de acceso a la información es muy importante. También lo es para el gestor de la infraestructura, pero éste también debe enfrentarse al problema de que el transporte de los datos de un data center a otro, con frecuencia muy alejados, supone un considerable consumo eléctrico que influye negativamente en la eficiencia del sistema.

paralelo y gestión de errores, todo cada vez más complejo, amenazaba con convertirse en un trabajo mayor que el que realmente interesaba realizar.
Se tomó la decisión de apartar a un lado todas estas tareas de gestión, incluyéndolas como librería del nuevo modelo de programación.

Por otro lado también se observó que la mayor parte de los cálculos realizados guardaban cierta similitud con las primitivas *map* y *reduce* presentes en varios lenguajes de programación funcional, como Lisp.
En esencia, los datos de partida se presentaban en forma de pares clave/valor, a los que se aplicaba una operación *map* que daba como resultado una serie de nuevos pares clave/valor intermedios. A este nuevo conjunto de datos se le aplicaba una operación *reduce* que los combinaba por sus claves, dando lugar al resultado buscado. Tratándose de un modelo de programación funcional, ambas operaciones, *map* y *reduce*, son escritas por el usuario.

Este modelo permite llevar a cabo gran cantidad de cálculos en paralelo de una forma sencilla. De hecho los valores intermedios se van entregando a la función *reduce* mediante un iterador, sin necesidad de disponer de la totalidad de los mismos: Esto permite manejar conjuntos demasiado grandes como para poder almacenarse en memoria.

Siguiendo la misma filosofía de reducción de la complejidad del modelo, se tomó la re-ejecución de tareas como el principal mecanismo de recuperación frente a errores.

Esta gran simplificación en las tareas reduce significativamente los requerimientos de sistema de las máquinas que han de ejecutarlas y hacen que el mecanismo MapReduce pueda funcionar óptimamente sobre PCs de propósito general sin más que aumentar o disminuir su número en función de la exigencias de datos de entrada y tiempos de respuesta requeridos.

MapReduce en acción

La implementación concreta de MapReduce variará en base a la infraestructura y máquinas de que se disponga.
El proceso siempre será iniciado por una aplicación del usuario, la cual será la encargada de lanzar todas las instancias del programa en el clúster de máquinas disponibles. Uno de los nodos será el *maestro*, el gestor del trabajo, que se encargará de la asignación de trabajos y supervisión del resto de nodos *esclavo* o gestor de tarea.

En primer lugar, la aplicación del usuario, a través del Sistema de Archivos Global, divide los datos de entrada en bloques de entre 16 y 64MB. Cada uno de estos bloques generalmente se almacena en más de una ubicación para proporcionar una mayor resistencia frente a fallos en los dispositivos. El gestor del trabajo registra la ubicación de cada bloque.

Se divide el trabajo en M tareas *map* y R *reduce*, donde se intenta que M y R sean bastante mayores al número de nodos, para mejorar el reparto de carga y tener una recuperación más rápida frente a fallos de algunos de ellos. También se intenta que M sea mucho mayor a R, con R siendo sólo unas pocas veces mayor que el número de

nodos. El gestor del trabajo asigna tareas (*map* o *reduce*) a cada uno de los nodos *esclavo* que estén libres.

Los gestores de tarea a los que se ha asignado funciones *map*, leen el bloque de datos que se les ha asignado. Para optimizar el uso de la red, siempre se intenta que dispongan en su misma máquina de dicho bloque. Si no fuera así o hubiese algún error en los datos, intentarán leerlo de la siguiente máquina más próxima (por ejemplo, conectada a su mismo switch), que disponga de ellos.

Según procesa los pares clave/valor de entrada, el gestor de tarea va almacenando en memoria los nuevos pares clave/valor intermedios, y periódicamente los almacena en el disco local, con R particiones. Cuando hace esto, pasa la información al nodo *maestro*, que a su vez pasará la información a los gestores de tarea *reduce* que vayan quedando libres.

En ocasiones puede ocurrir que las claves intermedias se repitan mucho y pueda interesar combinarlas en el disco local antes de que esta información sea requerida por tareas *reduce*. Esta función *combine* se ejecutará, por tanto, en la misma máquina que se ejecutó la función *map* que dio lugar a los pares clave/valor intermedios, y utilizará el mismo código que las funciones *reduce*. Es importante tener en cuenta que para que se pueda ejecutar una función *combine*, el código de la función *reduce* debe ser conmutativo y asociativo.

Cuando el gestor del trabajo pasa a una tarea *reduce* la ubicación de datos intermedios para su tratamiento, dicha tarea hace una llamada de procedimiento remoto (RPC) para recuperar los datos del disco duro local de la máquina que los contiene – de aquí la utilidad de la función *combine* en ocasiones para reducir la carga de esta transferencia. Una vez se cuenta con todos los datos en local, éstos se suelen ordenar, y cada clave con todo su conjunto de valores intermedios se pasan por la función *reduce*, cuya salida se va añadiendo a un fichero.

Todo este proceso se repite hasta que se completan todas las tareas *ma*p y *reduc*e, momento en el cual el gestor del trabajo devuelve el control a la aplicación del usuario. En este punto, la información resultante se encuentra dividida en R ficheros salida, que pueden pasarse por otra función MapReduce o a una aplicación que pueda tratar una entrada distribuida en múltiples ficheros.

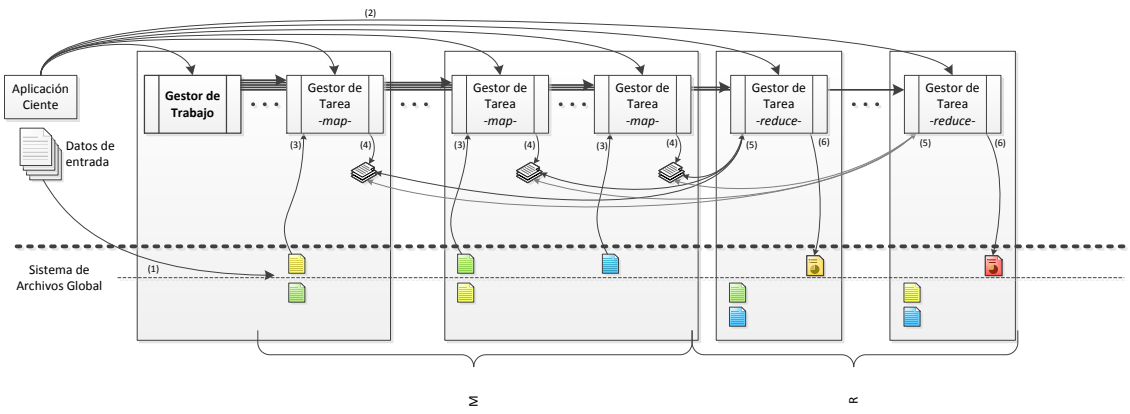


Figura 2. Funcionamiento MapReduce

Gestión de errores

Si estamos tratando con cientos o miles de máquinas de uso cotidiano, es habitual que encontremos errores. Como ya se anticipó, MapReduce basa la gestión de los mismos en una re-ejecución de la tarea que haya fallado.

El gestor del trabajo monitoriza todos los nodos periódicamente y si no recibe respuesta de alguna de ellos en un cierto lapso de tiempo, la marcará como fallido:

- Las tareas *map* del nodo, bien completadas o en progreso, vuelven a estado libre, con lo que pueden volver a escogerse para su ejecución. El hecho de que también se marquen las completadas viene del hecho de que sus pares clave/valor intermedios se almacenaban localmente en el nodo y ya no están accesibles para las tareas *reduce*. Las tareas *reduce* que tuvieran que leer del nodo fallido son notificadas del nuevo nodo que generará los datos intermedios que necesita.
- Las tareas *reduce* del nodo que se encuentren en progreso se marcan como libres, para poder ser asignadas y ejecutadas de nuevo. Al contrario que con las *map*, la salida de estas tareas se almacena en el Sistema de Archivos Global, que se considera libre de errores.

Aunque es más difícil que el gestor del trabajo falle, pues reside en un único nodo, puede ocurrir. En este caso es responsabilidad de la aplicación del cliente el gestionar dicho error. Puede optar por una re-ejecución completa o, si el gestor del trabajo almacenaba periódicamente las estructuras de datos y estado de las tareas, retomar desde el último punto almacenado.

Un caso especial de re-ejecución viene no de un error propiamente dicho. Lo constituyen los conocidos como *rezagados* (*straggler*). Se trata de tareas a las que completarse les lleva un tiempo mucho mayor del habitual. Los motivos para ello son múltiples: desde ejecución simultánea de otros procesos en el nodo hasta discos defectuosos con errores recuperables pero cuya velocidad de acceso ralentiza la ejecución.
Uno de los modos en que MapReduce puede tratar con estos *rezagados* consiste en asignar tareas aún en proceso a nodos libres cuando se aproxima la conclusión del trabajo. Así, la tarea se marca como completada cuando acaba su ejecución en cualquiera de los nodos que la están procesando.

Arquitecturas para el procesamiento de Big Data

MapReduce proporciona un modelo de programación para enfrentarse con Big Data sin perderse en la complejidad de gestionar el trabajo distribuido, pudiendo así centrarse el usuario en el auténtico problema que quiere resolver. Así, cuanto mayor es el conjunto de datos a tratar o la rapidez necesaria en la respuesta, basta con incrementar el número de máquinas acordemente. ¿O no?

Teorema CAP

En el año 2000 Eric Brewer formuló su teorema CAP para sistemas distribuidos, en el cual afirmaba que tales sistemas sólo podían asegurar dos de los siguientes tres aspectos:

- **C**onsistency: Tras una escritura en un nodo, la lectura en cualquier otro debe devolver dicho valor o uno aún más reciente (si se ha producido otra escritura).
- **A**vailabiliy: La disponibilidad debe ser tal que, al realizar una consulta a un nodo, proporcione una respuesta en un tiempo adecuado.
- **P**artition tolerant: Ante un corte entre los nodos (por un problema en la red, o fallo en las comunicaciones del nodo consultado), las garantías previas aún deben mantenerse.

De este modo, ante una lectura en un nodo, puede ocurrir que:

- Se devuelva inmediatamente el valor almacenado en ese nodo que, si en esos momentos hay un corte en la red, puede no ser consistente (por haberse actualizado el valor en otro nodo durante el corte). Sacrificamos consistencia en favor de disponibilidad.
- Dicho nodo, ante un corte de red, se quede esperando a verificar si el valor del que dispone es consistente. Sacrificamos disponibilidad en favor de consistencia.
- No exista ningún problema en la red y devuelva un valor consistente. Sacrificamos la tolerancia a cortes para contar con disponibilidad y consistencia.

Arquitectura Lambda

En 2011 Nathan Marz escribió una entrada en su blog en la que describía la manera de enfrentarse con la limitación impuesta por el Teorema CAP[viii].
Admitiendo que la limitación existe, identificó los aspectos que más aumentan la complejidad de enfrentarse a la misma: El uso de bases de datos mutables y algoritmos incrementales para su actualización.

Su aproximación al problema se basa en considerar los datos como entidades ligadas al tiempo y, por tanto, inmutables: Un dato es un hecho que es cierto en un momento determinado. De este modo pasa del concepto CRUD (Create, Read, Update, Delete) a sólo CR, los datos sólo pueden crearse y leerse, y se descarta la utilidad de su

- **Actualización**: Si se produce un cambio en los datos, está ligado a un nuevo momento en el tiempo, con lo que debe tratarse como la creación de un nuevo dato: Si una persona cambia de número de teléfono de contacto, no cambia el hecho de que previamente tuviera otro. Basta con consultar el último conocido.

- **Borrado**: De manera similar, que una persona se dé de baja de un servicio no cambia el hecho de que se diera de alta en una determinada fecha, y el hecho debería registrarse como una baja del mismo en la fecha correspondiente. De nuevo se crearía un nuevo dato[3].

Por otro lado, las consultas sobre los datos deben realizarse sobre la totalidad de los mismos.

Recordemos que el Teorema de CAP aún se aplica:

- Con prioridad de consistencia sobre disponibilidad, no hay cambios sobre lo ya conocido: ante un corte no se podrán escribir ni leer datos
- Con prioridad de disponibilidad sobre consistencia, sí se aprecia la ventaja de la aproximación de Marz al problema. En caso de corte, lo peor que puede ocurrir es que se proporcione un dato desactualizado que, eventualmente, se actualizará. Puesto que las consultas se hacen sobre el conjunto completo de datos cada vez y no de forma incremental, no se introducen errores y no son necesarios complejos sistemas para corregirlos una vez finaliza el corte.

Ahora bien, hasta ahora la premisa ha sido que es posible realizar una consulta sobre la totalidad de los datos, lo que en la práctica no es posible por restricciones de tiempo de procesamiento.

No obstante, en su razonamiento, Marz propone comenzar permitiendo que las consultas a los datos puedan estar desfasadas un cierto tiempo, unas horas por ejemplo, cosa que resuelve más adelante. ¿Cómo dar respuesta rápida a una consulta? Teniendo las consultas pre-realizadas, y volviéndolas a preprocesar cada vez que se añadan datos nuevos. Este preproceso, como es lógico lleva un tiempo considerable, y de ahí que haya que admitir proporcionar resultados algo desfasados.

Hadoop fue la herramienta escogida con la que enfrentarse al problema, pues consta de un sistema de ficheros distribuido (HDFS), que puede gestionar fácilmente un sistema de ficheros en constante crecimiento, y de un sistema de procesamiento por lotes fácilmente escalable para realizar consultas sobre la totalidad de esos datos, MapReduce.
Los resultados proporcionados por esta herramienta son entonces indexados en una base de datos simple que permita escritura por lotes (no aleatoria) y lectura aleatoria, que permita su rápido acceso desde la aplicación del usuario. Este conjunto es lo que denomina *Batch Layer* o Capa de Procesamiento por Lotes.

El funcionamiento de la *Batch Layer* no sólo permite, como vimos, que proporcione siempre resultados consistentes, sino que además se recupere fácilmente ante errores humanos. Los dos más frecuentes son:

- **Datos introducidos erróneos**: Como los datos son inmutables, basta con eliminar las entradas erróneas, pues estas no habrían sobrescrito valores previos, sino que se habrían añadido (*append*) al final de los existentes en el momento de su escritura. En otros sistemas esto no ocurre y, al compactar la base de datos, los

[3] Se acepta el hecho de que excepcionalmente pueda ser necesario el borrado de datos (por legislación o errores, por ejemplo), pero la solución propuesta puede tratarlo fácilmente.

valores antiguos se pierden. Una vez eliminado el error, la *Batch Layer* vuelve a procesar las consultas.

- **Consultas con errores**: Si las consultas contienen algún error en su implementación, basta con corregirlo y dejar que el sistema vuelva a reprocesar los datos con la versión corregida.

Para compensar el retraso en los resultados proporcionados por la *Batch Layer*, Marz propuso el uso de una *Realtime Layer* o Capa en Tiempo Real, que se ejecutase en paralelo con la primera cubriendo aquellos datos que no se recogieron desde su último preprocesado. En este caso, el sistema se basará en actualizaciones incrementales, que reducen considerablemente el coste computacional, y aunque su gestión suele resultar compleja, al tratar un conjunto relativamente reducido de datos, es más sencilla que si tuviera que procesar la totalidad de los mismos.
En cuando a la tolerancia a errores, aunque no se debe descuidar su corrección en la *Realtime Layer*, hay que relativizar su importancia y el esfuerzo a dedicar, dado que, eventualmente, serán corregidos cuando la *Batch Layer* alcance esos datos.

La arquitectura resultante propuesta por Marz, es lo que se conoce como *Arquitectura Lambda*. Aunque en su propuesta original no la abordó, dejando implícito que la aplicación del usuario debía combinar las consultas a las *Batch* y *Realtime Layers*, en general se acepta que la arquitectura contenga la *Serving Layer* o Capa servidora, que será la encargada de hacerlo ante la petición de la aplicación del usuario.

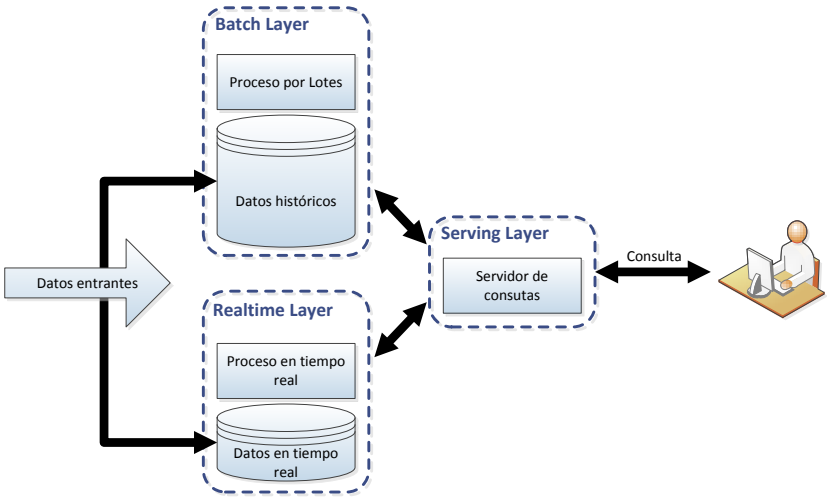


Figura 3. Arquitectura Lambda

Arquitectura Kappa

En 2014 Jay Kreps, de LinkedIn, publicó un artículo señalando la principal desventaja de la arquitectura Lambda y una alternativa a la misma, si bien reconocía el valor de mantener los datos históricos inmutables.

Expuso que mantener el código de dos sistemas distribuidos (*Batch* y *Realtime layers*) para que generasen el mismo resultado era una labor complicada, acrecentada por el hecho de que generalmente los dos sistemas se basaban en plataformas diferentes (pe. Hadoop y Storm).

Una alternativa que exploraron en LinkedIn fue el diseño de una API que permitiese programar de forma transparente para tiempo real y Hadoop para, de esta forma, sólo tener que diseñar un único código, dejando que la API lo ejecutase en ambos sistemas. Si bien consiguieron que funcionase, resultó extremadamente complejo dado que requería de profundos conocimientos de Hadoop, la capa en tiempo real y del funcionamiento de la propia API y como ésta traducía a cada uno de los sistemas. En consecuencia, esta alternativa fue descartada.

Como no pudo abstraer el empleo de dos sistemas diferentes, para así poder el diseñar y mantener una única aplicación de usuario, optó por prescindir de uno de ellos: la *Batch Layer*. Su premisa es que, si contaba con almacenamiento inmutable, los sistemas en tiempo real ya se encontraban lo suficientemente desarrollados como para ser escalados y poder procesar toda esa información.

Utilizando un sistema en tiempo real, sólo sería necesario reprocesar cuando el código cambie, y entonces lo único que habría que hacer es arrancar una nueva instancia del proceso en tiempo real que comenzase desde el principio de los datos disponibles, dirigiendo su salida a una nueva tabla. Cuando esta nueva instancia alcanzase a la anterior, se indicaría a la aplicación que puede leer de la nueva tabla, se detendría la instancia anterior y borraría su tabla de salida[4].

Esta nueva arquitectura es lo que se conoce como Arquitectura Kappa, y su principal ventaja es evidente: Su *simplicidad*. Sólo debe mantenerse un único código y no hay que combinar las salidas de dos sistemas independientes. Desafortunadamente, también tiene algunas implicaciones negativas sobre su eficiencia en el *rendimiento*:

- Necesidad de duplicar el espacio de la base de datos de salida durante el reprocesamiento.

- La base de datos debe ser capaz de soportar un régimen de entrada de datos muy elevado durante el reprocesamiento.

[4] Para el almacenamiento optó por Kafka que soporta replicación y tolerancia a errores en máquinas de uso cotidiano, pero en esta arquitectura habría podido emplearse cualquier sistema que tuviese capacidad de almacenar una gran cantidad de datos ordenados (como HDFS que utilizó Marz). Para el procesamiento en tiempo real empleó Samza, pero podría haber utilizado otros, como Storm.

Para Kreps, según las necesidades de la aplicación y capacidad de la infraestructura se debe elegir entre un sistema por lotes, uno en tiempo real o esta nueva arquitectura, pero siempre fue contrario a una arquitectura como la Lambda por la complejidad para los responsables de su desarrollo y mantenimiento.

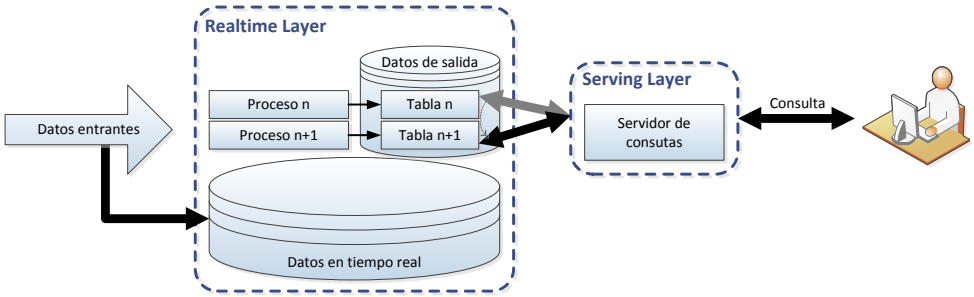


Figura 4. Arquitectura Kappa

Sistemas para tratamiento de Big Data

Las exigencias del Big Data varían en función de cada caso concreto dependiendo de los requisitos de capacidad, velocidad de respuesta o naturaleza de los datos. Esto ha dado lugar a la aparición de un gran número de soluciones en el mercado que han ido evolucionando para adaptarse a dichas exigencias y, en otros casos, ha llevado al desarrollo de nuevos modos de afrontarlas.

A continuación se presentan algunos de los productos más significativos, comenzando por los de código abierto y libre distribución que, por su naturaleza, permiten una visión más detallada de los componentes que constituyen una solución para el tratamiento de Big Data. Posteriormente se mostrarán algunos ejemplos de soluciones completas y específicas para el tratamiento de grandes cantidades de datos cuyo origen son registros (*logs*) de máquinas de diversas naturalezas.

Libre Distribución - Enfoques

Apache Hadoop

Apache Hadoop[5] fue el primer producto que ganó popularidad para el tratamiento de Big Data en la comunidad de código abierto, basado en los documentos y presentaciones de Google sobre como trataban las ingentes cantidades de datos en aquel momento. Su especialidad es el tratamiento de información por lotes, basándose en MapReduce como su motor nativo de procesamiento.

Se compone de múltiples capas o componentes:

- **HDFS** (Sistema de Ficheros Distribuido de Hadoop): Se encarga de coordinar el almacenamiento y replicación de los datos en los nodos del clúster, asegurándose de que éstos estén disponibles aun cuando se produzcan fallos en los equipos. En la configuración básica, se utiliza para almacenar la información de origen, resultados intermedios de procesamiento y los resultados finales.
- **YARN** (Aún Otro Negociador de Recursos): Su función es gestionar los recursos del clúster y programar los trabajos que deben ejecutarse.
- **MapReduce**: Motor de procesamiento nativo de Hadoop para proceso por lotes.

Su funcionamiento, basado en MapReduce, distribuye la carga de trabajo entre múltiples nodos en los que el principal requisito es la capacidad de almacenamiento, con lo que su coste es bajo. Esto, además, le permite manejar cantidades ingentes de datos. Por el contrario, su necesidad de frecuente lectura y escritura hace que sea bastante lento.

Se trata de un producto con una larga trayectoria y ampliamente probado, lo que le permite disponer de un amplio abanico de aplicaciones para ampliar sus funciones (o facilitarlas, pues MapReduce por sí solo requiere de unos conocimientos avanzados para poder ser utilizado) o ser tomado en cuenta para integrarse en otras arquitecturas que quieren hacer uso de sus capacidades.

[5] http://hadoop.apache.org/

Apache Storm

En contraste con Hadoop, Storm[6] se encuentra optimizado para el trabajo con flujos de datos en tiempo real, según se inyectan en el sistema, y potencialmente de tamaño ilimitado. Su latencia es mínima y garantiza que la información se procese al menos una vez. Sin embargo, en su versión más simple no garantiza que no se produzca duplicación de la información[7] ni que la misma se procese en orden.

Su funcionamiento se basa en Grafos Acíclicos Dirigidos (DAGs). Estos grafos se caracterizan por ser una representación de nodos y arcos en los que no existe un camino que comience y termine en un mismo nodo (ciclo), y en los que cada nodo tiene un valor topológico más bajo que el nodo siguiente, de modo que a lo largo de un camino por la topología, el orden se mantiene. Estos grafos suelen utilizarse en ciencias de computación para detectar cerrojos o *deadlocks* (puntos en los que los recursos deben esperar a que otro proceso finalice antes de continuar) o en estadística para modelar probabilidades. En el caso de Storm cada nodo representa una operación sencilla sobre un flujo de datos, y pueden estar conectados entre ellos o a fuentes de datos, conformando una topología que comprenda todo el procesamiento necesario.

Además de ser considerada una de las mejores soluciones para el tratamiento de la información en tiempo real, otra gran ventaja de Storm es su compatibilidad con una gran cantidad de lenguajes para poder definir las topologías de los DAGs (Perl, Python, Ruby, Javascript…), y su fácil integración con otros sistemas como YARN, que le permiten conectarse fácilmente a un entorno Hadoop ya existente.

Apache Spark

Spark[8] supone la evolución de una arquitectura de procesamiento de lotes destinada a acelerar el procesamiento mediante a) la optimización de los procesos y b) trabajar todo el tiempo en memoria (salvo en el momento de recuperar los datos para cargarlos y al escribirlos al final para su persistencia).

La optimización en el procesamiento se consigue, nuevamente mediante DAGs que representan todas las operaciones a realizar, los datos sobre los que realizarlas y las relaciones entre ellos. Con toda esta información conocida de antemano, el sistema puede coordinar todo el trabajo de forma óptima.

Para poder trabajar todo el tiempo en memoria, Spark utiliza Conjuntos de Datos Distribuidos Resilientes (RDDs): Estructuras de datos inmutables que representan grupos de datos. Cada operación sobre un RDD da lugar a otro RDD, y tienen la propiedad que puede remontarse al padre de cada uno hasta los datos en el disco. De este modo, ante un fallo, Spark puede recuperarse sólo con la información en memoria.

[6] http://storm.apache.org/

[7] Existen variantes de Storm, como Storm Trident, que garantizan que la información se trate una y sólo una vez, eliminando la posibilidad de duplicados. Sin embargo, para ello debe mantener información de estado lo que implica una importante variación del modo en que funciona Storm, pasando a micro-lotes en lugar de flujo de datos (como Apache Spark), y un aumento en la latencia.

[8] https://spark.apache.org/

El funcionamiento así descrito, está basado en trabajo por lotes, no por flujos. Sin embargo Spark también ofrece la opción de trabajar con flujos de datos de entrada. Esto no es estrictamente cierto, pero lo simula agrupando el flujo de datos en micro-lotes por intervalos de tiempo muy reducidos. Evidentemente, a mayor intervalo, más se aleja el rendimiento del funcionamiento en tiempo real. A cambio de este sacrificio, el sistema cuenta con un único motor para procesar tanto trabajo por lotes como flujos de datos.

El mayor inconveniente de Spark, sin embargo es su elevado consumo de RAM, pues almacena toda la información en memoria (en contraste con Hadoop, también basado en procesamiento por lotes), en lugar de en disco. Esto hace que el coste del hardware necesario sea mayor y que, en caso de escasez de recursos (memoria) pueda interferir con otros procesos que pudieran estar ejecutándose, en caso de ser equipos compartidos.

Apache Flink

Si Spark basaba su funcionamiento en un motor de procesamiento por lotes y adaptaba flujos de datos de entrada en tiempo real mediante micro-lotes, Flink[9] enfoca el problema desde el otro extremo: Utiliza un motor preparado para flujos de datos continuos y trata la información por lotes como un flujo de datos más, finito en el tiempo. Es una implementación de la arquitectura Kappa.

También en contraste con Spark, permite ciclos en sus diagramas de estado, mucho más eficiente a la hora de implementar algoritmos de Machine Learning, y gestiona aspectos como memoria o caché por sí mismo, dejando de lado otros mecanismos existentes.

Para resiliencia contra fallos, Flink almacena "fotografías" (*snapshots*) de los estados en ciertos puntos de la arquitectura, y para el almacenamiento puede configurar varios niveles de complejidad según la persistencia. Esto es así para el trabajo con flujos de datos pero en el caso de información por lotes, obvia las "fotografías" de estados, pues puede partir de los datos originales en caso de producirse un problema. De esta forma mejora el rendimiento.

Y es que Flink, que tiene como objetivo el procesado de la información en tiempo real, busca constantemente optimizar los procesos: Es capaz de distribuir tareas que puedan procesarse en paralelo y localizar las bloqueantes. Intenta realizar las tareas en los nodos donde se encuentran los datos o puede realizar *iteración delta* (iteración sólo en las fracciones de datos que han cambiado).

El mayor inconveniente de Flink es que es aún un producto relativamente joven, comparado con la mayoría, que todavía se está se está probando en grandes implementaciones.

[9] https://flink.apache.org/

Productos específicos para la gestión de logs

Como se verá en el siguiente apartado, el objetivo de este estudio es la evaluación de una arquitectura para gestionar los eventos generados en una red de comunicaciones con unas determinadas características. Por este motivo se muestran a continuación algunos de los productos de Big Data enfocados al análisis de logs, los cuales abarcan desde soluciones de código abierto a comerciales, como paquetes de instalación estándar o SaaS[10] en la nube.

ELK

ELK[11] es una solución de código abierto compuesta de tres componentes:

- **Elasticsearch**: Un motor de búsquedas capaz de almacenar y buscar en una gran cantidad de datos. Está preparado para funcionar en clúster, ocultando la complejidad para el usuario, dividiendo la información en *shards* (fragmentos), que puede distribuir en uno o más nodos (de forma óptima para repartir la carga de indexación y búsqueda), gestionando réplicas para aportar seguridad.
- **Logstash**: Utilizado para recolectar datos de fuentes heterogéneas, generalmente de forma pasiva (puede integrarse con *Filebeat* para la transferencia segura de logs), pero también proactivamente (Pe. Polling HTTP). Tiene, además, capacidad para añadir información o modificar la existente durante la recepción de los datos.
- **Kibana**: Para análisis y visualización. Permite crear paneles (*dashboards*) personalizados, con una gran variedad de representaciones gráficas (incluyendo geoespacial), y con capacidad de drill-down[12].

Se ofrece como un servicio en la nube o, de forma gratuita como código abierto, ofreciendo entonces la posibilidad de soporte de pago.

Uno de los problemas mayores de ELK es que, históricamente, Logstash ha tenido en su rendimiento y consumo de recursos su talón de Aquiles, siendo peores que los de otras soluciones equivalentes.

Graylog

Graylog[13] también se oferta como código abierto y, como ELK, usa Elasticsearch. Sin embargo usa MongoDB para el almacenamiento y Apache Kafka para mensajería. Existe una versión comercial que añade funcionalidades adicionales.

Se trata de un producto muy similar a ELK, con una interfaz gráfica y herramientas de gestión de usuarios (autenticación y permisos) superiores. Además, la obtención de informes resulta mucho más simple que con Kibana. Por el contrario, Logstash

[10] Software as a Service
[11] https://www.elastic.co/
[12] Consultar sección Visualización y Desglose: Drill-down.
[13] https://www.graylog.org/

proporciona a ELK una capacidad muy superior de ingestión de datos de diferentes naturalezas, y Kibana, con una curva de aprendizaje más lenta, puede llegar a ofrecer una interfaz de visualización más rica y personalizada.

Sumo Logic

Al contrario que los anteriores, Sumo Logic[14] se ofrece como un servicio comercial en la nube, capaz de recoger datos de múltiples fuentes estructuradas y sin estructurar, que puede etiquetar al ser almacenados, creando nuevos campos de metadatos. No precisa de una estructura predefinida o procesado de los datos para almacenarlos o indexarlos: Puede hacerlo manteniendo su formato nativo, para ser analizado más tarde en tiempo real. También destacan el empleo de Machine Learning para priorizar los datos que deben analizarse en primer lugar y en mayor profundidad, y el foco que ha puesto en la automatización y la posibilidad de crear scripts personalizados para cualquier clase de alerta.

Siendo un servicio en la nube, ofrece encriptación para mantener la seguridad de la información, potencialmente sensible, de las múltiples fuentes a la que accede. Sin embargo, también ofrece una variante híbrida en la que los recolectores de datos se instalan en máquinas Sumo en dependencias del usuario.

Tratándose de un producto comparativamente joven, adolece de ciertas carencias como una relativa escasez de funcionalidades, aplicaciones, documentación y soporte. Los básicos están cubiertos, así como la compatibilidad con los sistemas más importantes, pero no así con los aspectos más específicos o sistemas menos populares.

Splunk

Splunk[15] es una plataforma comercial para la recolección, búsqueda, monitorización y análisis de información basada en registros. Comenzó como un "Google para tus logs", pero su gran evolución le permite ahora almacenar toda clase de datos, realizar una gran variedad de análisis estadísticos sobre ellos y presentarlos en multitud de formatos.

Aunque recientemente lanzó una versión en la nube, tradicionalmente Splunk se ha venido instalando sobre máquinas del usuario, contando con instancias específicas para diversas tareas (Reenviadores, Indexadores, Cabezas de búsqueda) que permiten su despliegue de forma sencilla, versátil y escalable. Así mismo ofrece APIs para ejecutar consultas, introducir datos o integrarlo con otras aplicaciones (aunque posee su propio interfaz web de trabajo).

Su mayor virtud es su versátil Lenguaje de Procesamiento para Splunk (SPL), que permite a la vez modificar, filtrar, manipular, consultar o representar los datos indexados (También cuenta, al igual que Kibana, con *drill-down* integrado). Además su

[14] https://www.sumologic.com/
[15] https://www.splunk.com/

larga trayectoria en el mercado hace que disponga de gran cantidad de aplicaciones que extienden, aún más, sus ya amplias capacidades, así como una extensa documentación.

Similarmente a Sumo Logic, los datos en Splunk son almacenados sin estructuras específicas o procesamiento: El flujo de caracteres de entrada se divide en eventos en base a *timestamps*, y unos pocos metadatos adicionales, pudiendo añadirse a discreción del gestor de la plataforma otros campos indexados. Esto le permite minimizar los recursos para la ingestión de datos. La identificación y extracción de campos se realiza en tiempo de búsqueda.

Aunque resulta fácilmente escalable, el mayor inconveniente de Splunk es precisamente ese, el crecimiento: Su modelo de coste cobra al usuario por la cantidad de información diaria que se indexa. También ocurre con Sumo Logic, pero pone de relieve la importancia de este aspecto a la hora de tomar una decisión entre adoptar una solución comercial, mucho más preparada para un fácil y rápido despliegue, con sólo unos conocimientos básicos, pero con un coste económico importante, o una open source, que requerirá una inversión mucho más reducida, pero que requerirá de personal con experiencia y un período de tiempo mayor para su puesta en marcha.

A continuación se presenta una comparativa con la popularidad de las búsquedas en Google para las 4 plataformas presentadas en este apartado.

Figura 5. Comparativa de popularidad a partir de Google Trends

Análisis

El objeto de este estudio es el diseño de una plataforma que gestione los eventos en una red de comunicaciones con operaciones de ámbitos diversos, pero principalmente de tipo financiero y bancario. El origen de los datos es variado, procedente de logs de routers de diferentes fabricantes y máquinas Linux y Windows, así como registros específicos de las operaciones que atraviesan la red. El sistema debe ser capaz tanto de responder a consultas que se le realicen, permitiéndose un pequeño retardo (del orden de segundos), como de realizar un análisis continuo de los datos y generar proactivamente avisos frente a ciertas situaciones.

Datos de entrada

Naturaleza

Los datos de que disponemos provienen de una gran variedad de fuentes, pudiéndose dividir éstas en dos categorías según la clase de información que recogen:

- Funcionamiento de la red.
- Funcionamiento de las operaciones.

La red tratada en este estudio está compuesta por routers de diferentes fabricantes (Cisco, Juniper, Digi, etc), y de máquinas Linux y Windows que ejecutan aplicaciones específicas de la compañía. Los datos sobre el funcionamiento de la red provienen de los logs estándar enviados por los diferentes dispositivos que la constituyen (*push*) y de la información recogida de los mismos (*pull*), por ejemplo consultado sus MIB a través de SNMP periódicamente o bajo ciertas circunstancias.

Por otro lado, las aplicaciones de la compañía también generan registros con información de cada operación que atraviesa la red. El funcionamiento de las operaciones no depende únicamente de la red de la compañía, sino de factores externos tales como:

- **Redes** de terceros: Saturación, caída de circuito troncal, estación base o nodo RTC, cambio de enrutamiento a nueva ruta nunca antes probada (MTU inadecuada, servicios restringidos, …)
- **Hosts** en los que se entregan las operaciones: Mantenimientos, saturación, fallo eléctrico,…
- **Dispositivos** que generan las operaciones: Instalación de firmwares con errores, fallo físico en algún componente (modem, lector de tarjeta SIM…).

El correcto análisis de estos datos puede ayudar a detectar errores fuera de la red corporativa para actuar en consecuencia y proactivamente con los usuarios. Además permite la elaboración de informes de valor añadido para los clientes que no dispongan de la información o los medios para poder obtenerlos por sí mismos.

La diversidad de fuentes de información en la red trae asociada una heterogeneidad de los datos de entrada, si bien cada uno de ellos tendrá un timestamp asociado. La

mayoría de los datos ellos serán semiestructurados, y ocasionalmente podrán definirse nuevos campos para algunos de ellos.

Seguridad

Como se ha mencionado previamente, parte de la información que transita por la red bajo estudio es bancaria. Por este motivo los requisitos de seguridad son muy estrictos y, con el fin de asegurar la confidencialidad de las operaciones, existe información restringida (números de tarjeta, claves, etc.) que no debe ser registrada como parte de los datos. En general estos campos tan sensibles no forman parte de los datos de entrada al sistema de gestión de eventos, pero si en alguna circunstancia se incluyesen, el sistema debe ser capaz de enmascararlos, sin posibilidad de recuperación. Se prefiere esto sobre su eliminación para que, al menos, quede constancia de que el campo fue enviado.

Por otro lado, y aunque los datos críticos hayan sido enmascarados en el momento de almacenarse, la información que se está tratando es de carácter privado y el acceso a la misma debe ser seguro. Si se opta por un interfaz web para tal fin, deberá contar al menos con encriptación simple si los usuarios se encuentran en la misma red corporativa segura que el servidor. Si por el contrario los usuarios pueden acceder desde más allá de los firewalls de la empresa, deberán usarse certificados seguros.

La empresa necesita de la certificación PCI-DSS y está sometida a estrictas auditorías periódicas de seguridad. Entre sus requisitos figuran que cada persona que acceda a los dispositivos que conforman la red debe ser identificada y registrada inequívocamente. Por este motivo, extender la red a la nube como fuente de almacenamiento y procesamiento debe descartarse, y se deberá contar únicamente con los recursos corporativos.

Datos de salida

Visualización

En una red con miles de equipos como la que nos ocupa, se generaran millones de eventos diarios. Muchos de estos eventos serán irrelevantes o meramente informativos a efectos de gestionar la red, pero incluso ignorándolos, el remanente es imposible de gestionar a simple vista.

Es necesario disponer de vistas que aglutinen la información y la visualicen de una forma clara y concisa: Resúmenes con eventos concretos y número de repeticiones o valores promedios en el momento de la consulta resultan de gran relevancia en un sistema de este tipo: Equipo+Interfaz y número de caídas en los últimos 15min, o duración promedio de las operaciones autorizadas por cada host. MapReduce, como hemos visto, se adecúa a la perfección a esta clase de requisitos.

Yendo un paso más allá, también resultan de gran interés resúmenes que combinen varios eventos: Porcentaje de operaciones completadas con éxito respecto del total de operaciones contra cada host.

Aunque estos resúmenes aportan la información necesaria para que un usuario de la plataforma pueda detectar anomalías o eventos que merezcan un estudio en mayor detalle, visualizar tablas de números o mensajes de forma continua pronto provoca cansancio y no resulta viable. La inclusión de elementos visuales, que con un solo vistazo muestren la información, es indispensable.

Las gráficas de líneas que muestren la evolución temporal de los valores representados en las tablas antes mencionadas son una excelente solución, pues no sólo muestran el valor en el eje Y, sino que se puede comparar con sus valores previos (en X). Otra clase de gráficas como diagramas de barras o tarta también resultan de gran ayuda.

Finalmente, si la plataforma pudiera presentar la información visualmente más allá de gráficas simples, supondría una ayuda adicional no sólo para la detección sino para un primer diagnóstico de un problema.
Tomemos como ejemplo un caso en el que repentinamente se recibiesen alarmas de múltiples equipos simultáneamente: el usuario de la plataforma, intuitivamente, podría pensar en un problema eléctrico o con el proveedor de comunicaciones, pero tendría que investigar para poder profundizar en la posible causa. Sin embargo, si los equipos apareciesen representados en un mapa, coloreados en verde o rojo según su estado, y apareciesen concentrados en un área geográfica extensa, podría indicar un problema en una de las centrales del proveedor de comunicaciones.
Otros ejemplos de representaciones visuales más complejas podrían ser diagramas de red con las conexiones físicas entre equipos, o de los servicios en cada host.

Tiempo de respuesta

Puesto que los destinatarios de esta plataforma son personal de la empresa propietaria de la misma, y no público en general, se tolera un pequeño lapso en el tiempo de respuesta del sistema, de varios segundos.

Sin embargo, es necesario trabajar con datos actualizados, pues es esencial detectar los eventos importantes tan pronto se produzcan. Un funcionamiento exclusivamente basado en la *Batch Layer* de la *Arquitectura Lambda*, por tanto, no es tolerable.

Consultas sobre los datos

Si la visualización es importante para la detección de problemas, es crucial que el sistema facilite la consulta de los datos para llegar al origen de las mismas.

Ante la observación de una alarma o anomalía en las gráficas o tablas proporcionadas por el sistema, un experto en la arquitectura de la red tendría los conocimientos necesarios para saber los siguientes datos a comprobar y, en última instancia, los equipos que debe verificar y qué elementos dentro de ellos.
Sin embargo, aunque deseable, no siempre se puede contar con la disponibilidad de tal personal en todo momento que, por otro lado, suele reservarse como segundo nivel ante problemas de cierta importancia. Por este motivo, debe contarse con una buena documentación a la que remitirse en estas situaciones. Esta documentación indicará los pasos a seguir, hasta llegar a determinar finalmente a los elementos a verificar y posibles valores y/o estados esperables. Es crucial en este punto que los datos que han dado lugar a la visualización estén plenamente disponibles.

Tomemos como ejemplo una serie de marcadores en un panel de monitorización que proporcionen el porcentaje de operaciones completadas con éxito para cada cliente.
Si en un momento dado esta tasa cae por debajo de sus valores habituales será lícito pensar que existe un problema. El operador que ha detectado el problema deberá verificar si se presenta únicamente para un cliente o para varios, si ocurre por un circuito concreto o por múltiples, si lleva asociado otros aspectos como aumento o reducción en los tiempos de las operaciones, etcétera.
Su problema no radica tanto en saber conceptualmente qué debe verificar (sin haber descrito arquitectura todavía, las comprobaciones anteriores parecen lógicas), sino los registros concretos que se corresponden con ese cliente y que debe recuperar del sistema para su análisis.

A la vista de lo anterior, es indispensable que la plataforma proporcione acceso directo a la información sin procesar, mediante una herramienta de búsqueda potente y versátil, para poder llegar al origen de la anomalía. Por otro lado es deseable que la herramienta permita llegar a esta información de una manera intuitiva e inmediata, que no requiera de conocimientos específicos de cada escenario. Para ello resultará de gran utilidad la funcionalidad de *drill-down*, o *desglose*, ofrecida por Splunk, como se verá más adelante (sección Visualización y Desglose: Drill-down).

Generación de avisos

En el apartado referente a la visualización ya se hizo mención a la fatiga que lleva asociada una jornada completa de monitorización de la red. Esta puede desembocar en un posible fallo humano, bien por omisión en la detección, bien por una valoración/priorización errónea del problema.

Para dar respuesta a esta contingencia, la plataforma debe ser capaz de generar avisos de forma proactiva. Generalmente esta clase de avisos consistirá en correos a personal o departamentos relevantes para la atención y para la gestión del problema (también podría ser necesario informar proactivamente a clientes de la situación), aunque también puede tratarse de generar eventos que a su vez informen a otro sistema.

Así mismo, es importante que la plataforma tenga la capacidad de gestionar estos avisos en el tiempo, definiendo si deben enviarse una única vez, o periódicamente y bajo qué circunstancias.

Arquitectura de red

La red de comunicaciones para la cual se quiere implementar la plataforma de gestión de eventos tiene presencia en países de todo el mundo, con conexiones privadas, seguras y redundadas entre sus nodos. La red se encuentra dividida en una subred de gestión y otra dedicada propiamente a las comunicaciones provistas a los clientes, ambas separadas entre sí por motivos de seguridad y privacidad. Disponer de una red de gestión así de robusta facilita la implementación de una solución basada en equipos propios sustentada por la propia red, sin necesidad de recurrir a una solución en la nube.

Para facilitar su gestión, la red se encuentra dividida en tres grandes áreas geográficas que, aunque conectadas, pueden funcionar de manera independiente:

- Región 1: América
- Región 2: Europa
- Región 3: Asia-Pacífico

La arquitectura de red de cada una de estas áreas es similar a las de las demás, de modo que la descripción de una de ellas puede aplicarse al conjunto.

Los elementos más importantes de que consta la red pueden clasificarse según su función en:

Equipos de acceso

Permiten el acceso a la red de dispositivos de clientes para proporcionarles conectividad con hosts de otras entidades. Los dispositivos de clientes pueden ser datáfonos, concentradores de tráfico u otros hosts, y las tecnologías de acceso muy diversas: RTC, X.25, PPP, IP.

Estos equipos son máquinas Linux que ejecutan un software propietario de la compañía y generan información de tres tipos:

- **NMS** - Network Monitoring System: Mensajes generados por el software propietario de la compañía, que proporcionan información acerca de su estado, alarmas y eventos más relevantes. Cada equipo establece una conexión permanente con, generalmente, dos servidores NMS (para disponer de redundancia), a través de la cual envía estos mensajes. A su vez, los servidores NMS monitorizan que los equipos mantengan la conexión, y generan un mensaje de alarma con la pérdida de la misma, que se repetirá periódicamente mientras se prolongue.

- **Syslog**: La mayor parte de los mensajes de syslog, propios del sistema operativo, no se consideran relevantes y quedan registrados exclusivamente en la máquina que los genera. Sin embargo las facilities[x] 4 y 10, referentes a los mensajes de seguridad y autorización, dada su importancia, sí se envían a unas máquinas de syslog centrales para su gestión. El software propietario también hace uso de las facilities 19 y 20, de propósito general, para registrar ciertos eventos.

- **Contabilidad**: Cada operación que pasa por estos equipos deja un registro con información sobre la misma. Esta información depende del tipo de operación y los equipos de acceso y entrega. En general, y como mínimo, describe la operación sin atender a su contenido: Timestamp, duración, bytes transmitidos en cada sentido, equipo de entrada, código de finalización, etc. También es posible, bajo permiso de los clientes, acceder a otra información contenida en el mensaje, como identificador de terminal o de comercio. Bajo ninguna circunstancia, sin embargo, la información más sensible (Como números de tarjeta, nombres de usuario o importes de compras) queda nunca registrada.

Toda esta información se transmite a los respectivos equipos concentradores en tiempo real, si bien no existen mecanismos para recuperarla de forma automática en caso de pérdida de comunicación.

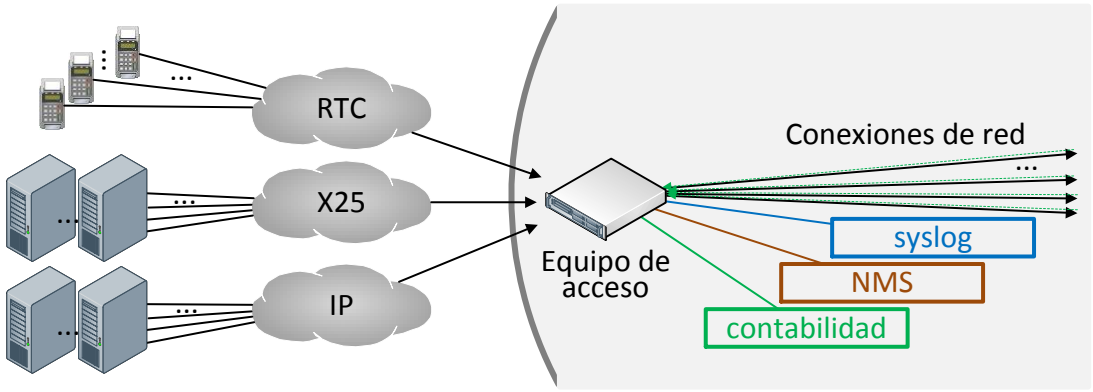


Figura 6. Equipo de acceso

Equipos de entrega

Al igual que los equipos de acceso, los equipos de entrega son máquinas Linux que también ejecutan un software propietario de la compañía. Su misión es adecuar los mensajes que los atraviesan al protocolo de comunicación y formato que espera el host al que se van a entregar.

Generan información de tipo NMS y Syslog. También pueden extraer datos de los mensajes que los atraviesan (Identificador de terminal o de comercio, etc), pero no generan un registro con dicha información, sino que se la envían de vuelta al equipo de acceso que les entregó el mensaje, para que este la añada a su registro de contabilidad.

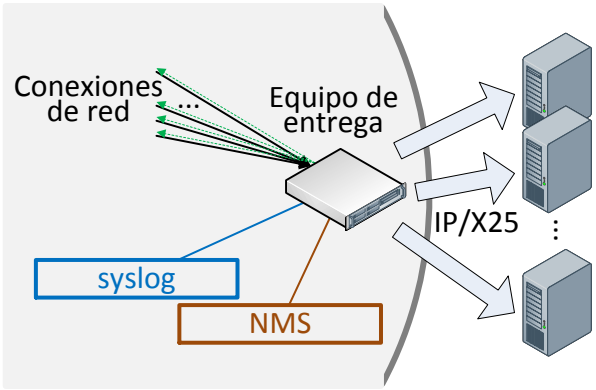


Figura 7. Equipo de entrega

Equipos de distribución

Los equipos de acceso se basan en características de la llamada entrante (Sea RTC, IP u otras tecnologías) para enviar el tráfico a un equipo de entrega u otro. El enrutamiento convencional de este tráfico dentro en la red se realiza a través de routers estándar (Cisco o Juniper). Sin embargo, en ocasiones la distribución del tráfico necesita de un análisis del contenido del mensaje para escoger el equipo de entrega adecuado. En estos casos, los equipos de acceso mandan el tráfico a unos equipos especiales de distribución, que acceden a la parte del mensaje específica y, en base a su valor, lo redirigen al equipo de entrega que corresponde.

Los equipos de distribución, como los de entrega, generan información de Syslog y NMS. A diferencia de los mismos, no sólo no generan registros información de contabilidad sino que tampoco extraen información de este tipo para enviar de vuelta a los equipos de acceso, sino que se limitan a redirigir esta información entre unos y otros.

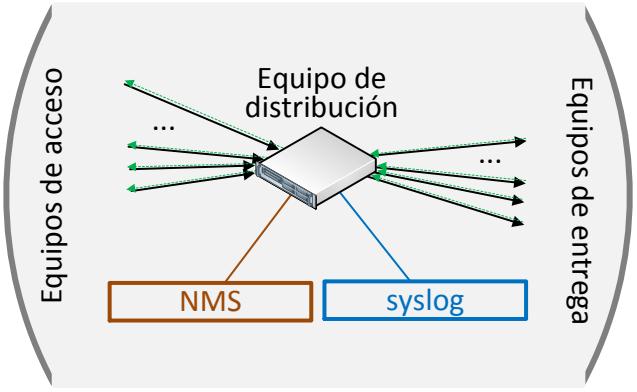


Figura 8. Equipo de distribución

Routers

Agrupar los routers de una red de comunicaciones como la que nos ocupa según su función, aunque posible, daría como resultado un listado tan extenso como inservible a efectos de este estudio. Baste decir que podemos encontrarnos con routers "desplazados" y routers en la propia red.

Los routers dentro de la red, cuyas funciones van desde la interconexión de las diferentes sedes de la compañía, pasando por la conectividad con proveedores y clientes, hasta la concentración de tráfico de "desplazados", mandan sus eventos al gestor de Syslog para su gestión. Además de esto, la compañía cuenta con una herramienta llamada Solarwinds, que le permite consultar proactivamente (SNMP) el estado de los routers de forma periódica para obtener información actualizada de su estado (MIB).

Los routers "desplazados", por otro lado, se encuentran fuera de la red corporativa y se conectan a la misma generalmente mediante túneles. Son únicamente consultados por Solarwinds y carecen de conexión Syslog.

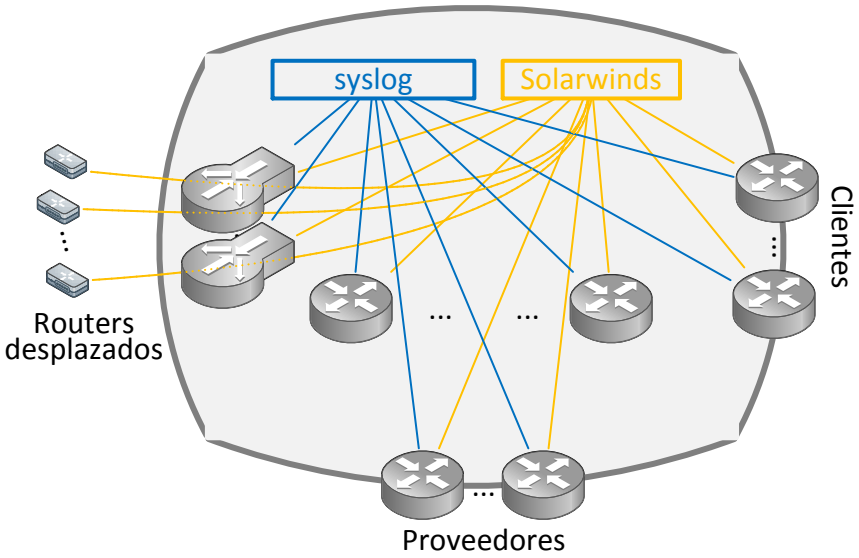


Figura 9. Routers

Servidores

Dentro de esta categoría se incluyen todos los equipos que no pueden clasificarse dentro de las anteriores. Generalmente se trata de equipos de gestión (syslog, NMS,…) o de propósito específico (TACACS, Radius,…). La información que de ellos se recoge es de dos tipos:

- **Estado del Sistema**: Se recoge información estadística del estado de los procesos que se ejecutan en estos equipos, como uso de CPU y memoria, para verificar su correcto funcionamiento y poder adelantarse a posibles problemas que pudieran surgir.

- **Autentificación**: En una red donde la seguridad es esencial, los equipos que gestionan Radius y TACACS manejan información tan sensible como es el acceso a los elementos y recursos de la red. Por este motivo, dicha información se trata de forma independiente del resto de información genérica de sistema que se recoge de estos equipos.

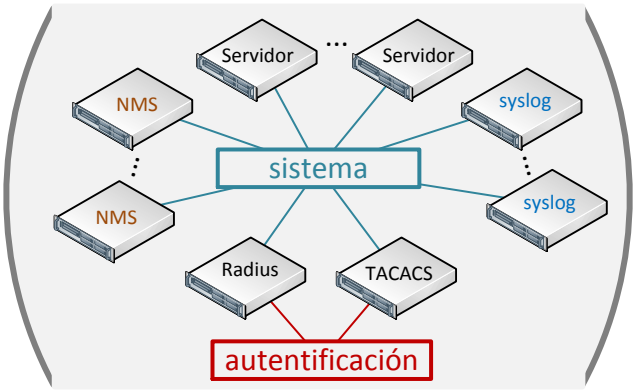


Figura 10. Servidores

Perfil de tráfico

La información a procesar, según su distribución en el tiempo puede clasificarse en tres categorías:

- **Uniforme**: Chequeos proactivos y periódicos de los equipos de la red para conocer su estado a través de diferentes parámetros consultados en los mismos. Se trata de un volumen de información constante en el tiempo.

- **Aleatorio**: Alarmas generadas proactivamente por los equipos de la red ante eventos relevantes. La red cuenta con un número de equipos muy elevado y asumiendo que, en general, su probabilidad de fallo es independiente, podríamos aceptar un volumen de alarmas uniforme en el tiempo. Sin embargo, la asunción anterior sólo es una aproximación para el funcionamiento habitual. En la realidad, debemos considerar que existen eventos, tanto internos con ajenos a la red, que pueden ocasionar la generación de alarmas en múltiples equipos (con lo que la probabilidad de fallo realmente no es independiente). Tal es el caso, por ejemplo, de la caía de un circuito o equipo troncal que aísle una sección de la red o la interconexión con un operador externo.

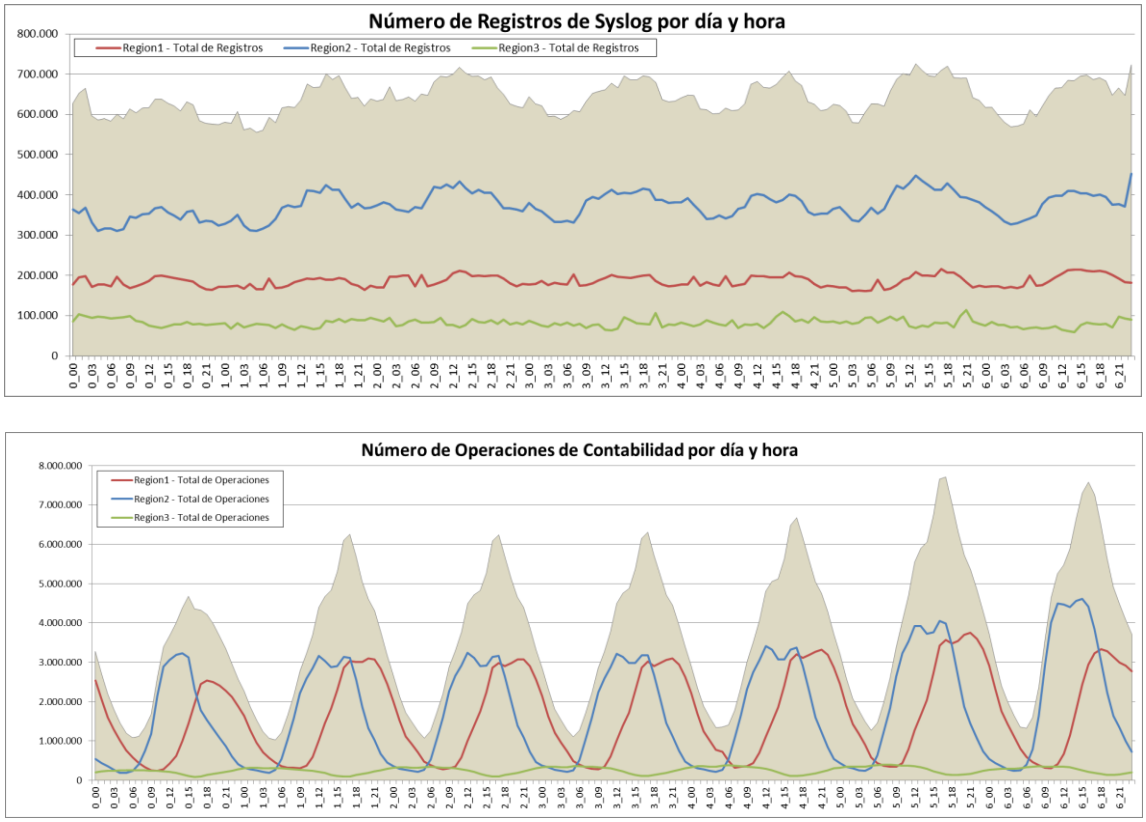


Figura 11. Distribución uniforme y cíclica del tráfico

- **Estacional/Cíclica**: Información dependiente del uso de los equipos y recursos de la red. El ejemplo más importante es la información de Contabilidad, donde cada operación de los clientes deja como resultado un registro de la misma. El volumen de información se encuentra fuertemente ligado a la hora y día de la semana y, algo menos, al mes del año.

Es importante tener en cuenta que en este comportamiento, dependiente de la hora, también tendrá importancia la ubicación geográfica donde se generan los eventos. En un instante concreto, puede ser hora pico en Madrid, pero valle en Hong Kong y también pico en Nueva York. Por este motivo, a la hora de dimensionar el sistema habrá que tener en cuenta este aspecto.

Resumen de requisitos

De lo visto en esta sección, se concluyen los siguientes requisitos:

ID	Requisito	Prioridad
R01	Etiquetado información por naturaleza/origen	Primario
R02	Capacidad importación información de diferentes naturalezas	Secundario
R03	Seguridad – Enmascarar información sensible	Secundario
R04	Seguridad – Ejecución en máquinas del cliente	Primario
R05	Seguridad – Definición usuarios/perfiles	Primario
R06	Representación gráfica básica – Tablas, líneas, barras, tartas	Primario
R07	Representación gráfica avanzada	Secundario
R08	Acceso rápido a información fuente de las gráficas	Primario
R09	Alertas	Primario
R10	Tiempo de respuesta	Secundario
R11	Escalabilidad	Primario
R12	Facilidad de implantación y gestión	Primario
R13	Soporte	Primario
R14	Precio	Secundario

Tabla 1. Requisitos del producto

La prioridad *secundaria* denota una característica deseable aunque no esencial:

- R02 y R03 pueden suplirse por scripts previos a la importación de los datos en el sistema.
- R07 puede sustituirse, en parte, empleando un mayor número de representaciones básicas en lugar de una única representación avanzada.
- R14: La empresa está dispuesta a pagar por un producto a cambio de facilidad de uso y soporte. No contempla una herramienta con un largo ciclo de aprendizaje de cara a su despliegue y posterior gestión, que suponga personal con dedicación exclusiva y de difícil reemplazo.

Si bien los productos presentados en el apartado Productos específicos para la gestión de logs satisfacen la mayoría de los requisitos técnicos contenidos en la Tabla 1, presentan algunas carencias:

- En el caso de Sumo Logic, no cumple con el requisito de seguridad R04, indispensable: Pese a disponer de una solución híbrida, parte de la solución aún se encuentra en la nube y aunque encripte la información, esta abandona la red de la empresa, no pudiendo garantizarse el cumplimiento del estándar PCI-DSS.

- ELK, que se ofrece como servicio en la nube fallando entonces también en R04, salva esta limitación por disponer de una solución open source para ejecutarse en máquinas del cliente. Sin embargo esto entra en conflicto con R12, pues pese a disponer de soporte de pago, no es un producto fácilmente desplegable. Además, su capacidad de gestión de usuarios es limitada (R05)

- Graylog a priori cumpliría con los requisitos primarios, pero su capacidad de representación gráfica es limitada (R07), así como la variedad de información que es capaz de *ingerir* (R02)

Splunk, por el contrario, satisface todos los requisitos, si bien lo hace a costa del precio de su licencia, más elevado que el de los otros productos. Este será el producto escogido para el diseño e implementación de la solución.

Diseño

La red de comunicaciones para la que se quiere implementar la plataforma de gestión de eventos está desplegada a escala mundial y, como se ha venido mencionando, se encuentra dividida en tres grandes regiones, siendo todas ellas similares entre sí. Los planteamientos y soluciones propuestos para una de ellas, será aplicable al resto.

Arquitectura Splunk

Splunk permite un despliegue personalizado según las necesidades particulares de cada caso: Desde una única instancia para gestionar la información de un departamento aislado, hasta un despliegue en múltiples máquinas de la red, escalable en base al volumen de información.

En el primer caso, la instancia Splunk instalada es una versión completa de la herramienta, con todas sus funcionalidades disponibles: captura, envío, indexación y búsqueda de información. Para el segundo supuesto, aunque sería posible el despliegue de instancias completas de Splunk en cada máquina, en la práctica esto no aportaría ventajas significativas. La herramienta permite la instalación de instancias con sólo una parte de las funcionalidades, lo que hace que sean más "ligeras" para el sistema. Así, podemos contar con:

Reenviadores (Forwarders)

En su formato básico (reenviador universal), estas instancias se ocupan únicamente de reenviar a los indexadores los datos que consumen de una o varias fuentes, lo cual llevan a cabo mediante la monitorización de ficheros o directorios completos, comunicación TCP por un puerto, o eventos SNMP o HTTP/HTTPS.

No alteran la información en medida alguna (si ello fuera necesario, debería acudirse a los Reenviadores Pesados - o Heavy Forwarders - con más funcionalidades que los universales, pero también de mayor tamaño y consumo de recursos), salvo por añadir los metadatos básicos de Splunk (Fuente, Tipo de Fuente y Host).

Para llevar a cabo el envío de la información, ofrecen una serie de opciones para configurarlos según las necesidades de cada caso concreto, como balanceo de carga, buffering, compresión, cifrado, clonado de la información o enrutamiento de la misma en función de su naturaleza.

Indexadores (Indexers/Peers)

Reciben la información de los reenviadores y son los encargados de realizar la indexación de la misma. La información se almacena comprimida, en bruto (*raw*), lo cual, para unos datos de entrada en formato texto como, por ejemplo, los procedentes de un syslog, puede suponer una disminución de un 85% de espacio ocupado en disco. Sin

embargo, los ficheros con índices creados por esta instancia y que hacen referencia al fichero en bruto, son considerablemente mayores (En el caso anterior pueden suponer un 35% del tamaño original de los datos)[16].

También se ocupan de llevar a cabo las tareas de búsqueda recibidas desde las cabezas de búsqueda para completar las peticiones de los usuarios.

Finalmente, cuando se trabaja en clúster, los indexadores se ocupan de mantener la información de respaldo sincronizada a la vez que indexan aquella que les llega desde los reenviadores. Al configurar el clúster en Splunk, se definen lo que se conoce como el factor de réplica y el factor de búsqueda, que no son otra cosa que el número de copias de datos en bruto y de ficheros de índices que mantiene el sistema para asegurar la redundancia de la información y protegerse frente a fallos eventuales de alguno de los nodos indexadores. En general suele escogerse un factor de réplica mayor pues, como se ha indicado, los ficheros de datos en bruto ocupan significativamente menos que los de índice, que en caso de extrema necesidad, podrían volver a re-generarse.

Cabezas de búsqueda (Search Heads)

Estas instancias se encargan de gestionar las búsquedas entre los indexadores, realizando peticiones a los mismos y consolidando los resultados devueltos.

Cada indexador almacena la información (raw y de índices) distribuida en archivos contenidos en directorios denominados cubos (*buckets*). Si el indexador pertenece a un clúster, los cubos se replican para cumplir con los factores de réplica y búsqueda. Para evitar duplicidades a la hora de buscar, una única copia de cada cubo en el clúster se marca como Primaria. Si el nodo en que se encuentra dicho cubo cayese por cualquier motivo, otra copia de cada uno de sus cubos en otros nodos se marcaría como primaria.

Al realizar la búsqueda en un clúster, la cabeza de búsqueda consulta periódicamente al nodo Maestro para conocer cuáles son los indexadores activos, estando así en disposición de comunicarse con ellos directamente. La cabeza de búsqueda mandará la misma petición a cada indexador, y será cada uno de ellos quien decida si debe realizar la búsqueda, en función de los cubos primarios de que disponga.

Las cabezas de búsqueda, además, mantienen actualizados en los indexadores los llamados paquetes de conocimiento – o *knowledge bundles* –, los cuales contienen los elementos necesarios para que los indexadores puedan realizar las búsquedas en su nombre.

De igual modo que existen clústeres de indexadores, pueden definirse clústeres de cabezas de búsqueda.

[16] Mediciones prácticas en el apartado 'Inyección de datos'

Maestro (Master)

Existe una única instancia Maestro en un clúster, y es la encargada de coordinar el funcionamiento de todas las demás: Coordina la replicación de los datos e índices de los indexadores para cumplir con los factores de réplica y búsqueda, gestiona la eventual caída de nodos dentro del clúster, informa a las cabezas de búsqueda de donde se encuentran los datos, etc.

Así mismo, el nodo Master permite distribuir información de configuración común entre los diferentes nodos, para mantenerla homogénea y actualizada con facilidad.

Al no existir ninguna otra instancia que cubra las funciones del Maestro, en caso de que este nodo falle, el resto del clúster podrá continuar funcionando durante algún tiempo sin incidentes, confiando en la última información que les hizo llegar el Maestro. Sin embargo eventualmente los resultados devueltos dejarán de ofrecer garantías, máxime si se produce la caída de algún otro nodo o se generan nuevos cubos durante este estado inconsistente del sistema.

En el caso que nos ocupa en este estudio, optaremos por un despliegue pequeño, pero capaz de incrementarse sin que ello suponga grandes cambios en el diseño. Por este motivo se instalarán los cuatro tipos de instancias antes descritas, de modo que si se necesita aumentar la disponibilidad o capacidad de la arquitectura, sólo haya que aumentar su número pero no alterar su estructura. Nos centraremos en el estudio de la información de contabilidad, dado el importante porcentaje que representa sobre el total de datos generados en esta red de comunicaciones, así como por las peculiaridades que presenta frente a otros logs más estándar.

La Figura 12 muestra la arquitectura propuesta para la gestión de los eventos de nuestra red de comunicaciones. La red existente ya fue descrita en la subsección *Arquitectura de Red* dentro del *Análisis* de los requisitos de la plataforma.

En cada región mundial, los equipos que recogen información se encuentran redundados, estando cada pareja formada por uno *Principal* y otro de *Respaldo*. Ambos almacenan toda la información, pero el *Principal*, además, se utiliza para alimentar las herramientas de monitorización actuales de la red. En todos estos equipos se instalarán las instancias *forwarder* de Splunk, si bien en una primera fase deberían activarse únicamente en los *Respaldos* de cada pareja para verificar su funcionamiento e impacto sobre el rendimiento del sistema. En caso de fallo de la máquina, se podría activar manualmente el *forwarder* del equipo *Principal*.

Los *forwarders* de cada región mundial harán llegar la información a una pareja de *indexers*. Finalmente los *indexers* serán accesibles desde tres *search heads*, una en cada región.

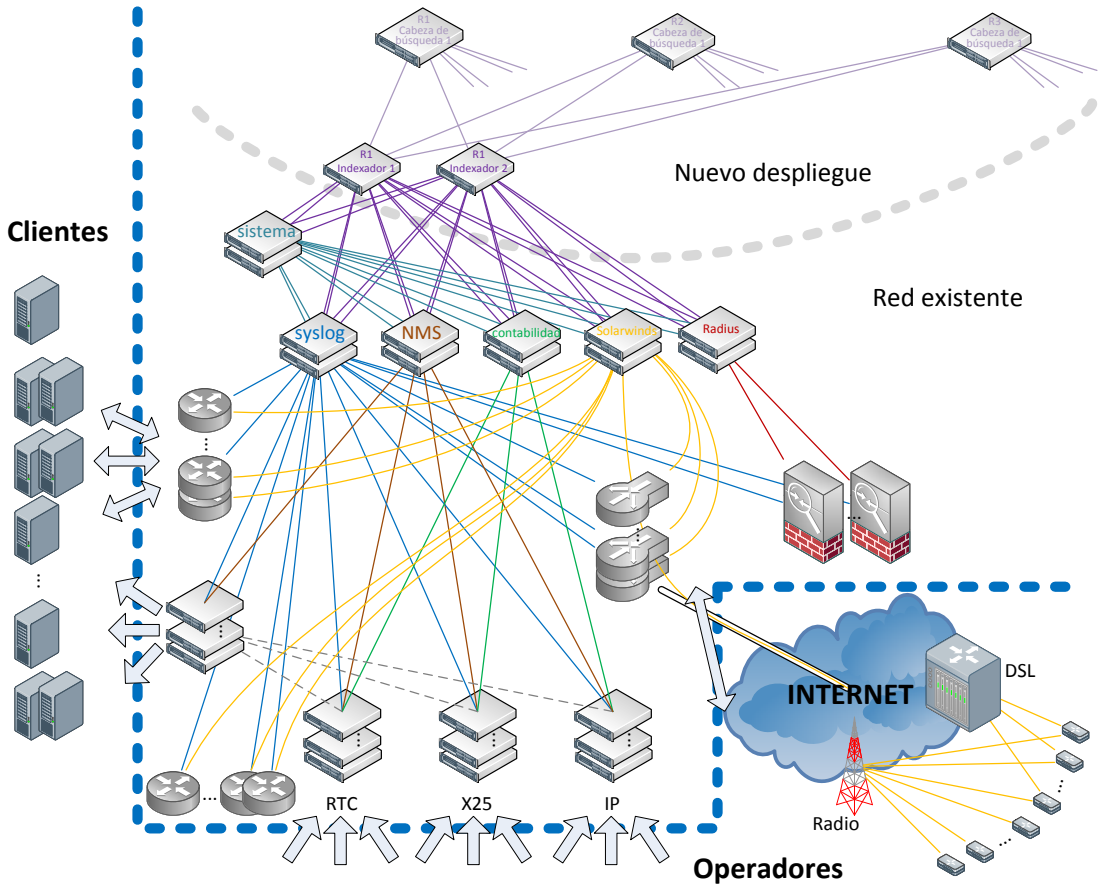


Figura 12. Arquitectura propuesta

Los *forwarders* estarán instalados en máquinas ya existentes, pero tanto los *indexers* como las *search heads* deberán desplegarse en nuevas máquinas dedicadas a esa función específica. Esto significa que se requerirán 10 máquinas nuevas: 2 *indexers* y 1 *search* head por cada región mundial, además de un *Master*.

Pre-procesado

Splunk se encuentra preparado para poder importar en sus índices información de cualquier tipo, bien con estructura estándar (pe.syslog), bien con estructura propietaria. Este último es el caso de la información de contabilidad de la red en estudio. Es más, la información se encuentra codificada de diferentes modos dependiendo del equipo que la genera y, aunque comparten ciertos campos comunes, no todos proporcionan los mismos datos.

Se plantea la posibilidad de realizar un pre-procesado de la información en las máquinas que recogen la información de contabilidad, de forma que se unifique el formato antes de enviarse a los procesos *forwarder* residentes en las mismas. Esto presentaría dos ventajas con respecto a definir las estructuras de datos en Splunk para que los *indexers* interpreten correctamente los eventos.

En primer lugar, la carga de la transformación de la información se recaería en unas máquinas a priori con poco uso, en contraste con los *indexers*, que deben indizar la información proveniente de todas las fuentes de la red (aun siendo muy extensa, la información de contabilidad no es la única).

En segundo lugar, y más importante, se encuentra el aspecto del mantenimiento de la herramienta. Ante una modificación en los formatos de contabilidad, debería modificarse de forma acorde la definición de la estructura de datos en Splunk para poder indexarlo correctamente. Por suerte, los mismos desarrolladores del software encargado de generar la información binaria de contabilidad, mantienen a su vez actualizada una pequeña aplicación que permite consultarla en formato de texto. Utilizando esta herramienta podría extraerse la información de las diferentes fuentes de contabilidad y unificar sus formatos, sin necesidad de realizar cambios en los *indexadores* de Splunk con cada nueva versión de los generadores de información de contabilidad.

Información disponible

En la Tabla 2 se muestra un resumen de la información de contabilidad disponible para cada tipo de fichero generado por los equipos que la registran. Los tipos de fichero 1-3 son más antiguos y su estructura es fija, con los valores de los campos en posiciones específicas del registro y tamaño fijo. El tipo 4, más moderno, es más flexible y permite que se puedan añadir nuevos campos sin más que asignarles una nueva etiqueta que los identifique. De hecho, el tipo 4 de ficheros dispone de muchos más campos que los que muestra la tabla, pero que no se han incluido por no estar disponibles para ninguno de los otros tipos de fichero.

El pre-procesado que se plantea consistiría en tomar los datos provenientes de los diferentes tipos de fichero y convertirlos en nuevos registros con pares clave=valor que se importarían en Splunk:

- Tipo 3: Sería el más sencillo de importar. Tiene un número fijo de campos, siempre en el mismo orden, con los que los pares clave=valor se pueden establecer directamente.

- Tipos 1,2: Son iguales al tipo 3, pero con una particularidad. Para salvar las restricciones en el número de campos ante nuevas necesidades que surgieron en la compañía, se optó por modificar la información introducida en algunos de ellos: En ocasiones yuxtaponiendo información adicional (Caso del tipo 1, donde , si está disponible, el valor del campo “Puerto” se añade al campo “Dirección”), y otras veces sustituyendo completamente el valor de un campo por otro diferente (En el tipo 2 puede ocurrir que se sustituya en ocasiones el “Id_Origen” por el valor de un “Puerto”). Por este motivo se deberá identificar cada tipo de registro durante el pre-procesado para extraer, si procede, la información correctamente de los campos modificados y generar las parejas clave=valor adecuadas.

	Tipo de Fichero						
Campo	1	2.a	2.b	2.c	2.d	3	4
Fecha_Inicio	X	X	X	X	X	-	X
Fecha_Fin	-	-	-	-	-	X	O
Id_Formato	-	X	X	X	X	X	O
Id_Servicio	-	-	-	-	-	X	-
Tipo_Servicio	-	X	X	X	-	-	O
Duración	X	X	X	X	X	X	O
Bytes_de_host	X	X	X	X	X	-	O
Bytes_a_host	X	X	X	X	X	-	O
Id_Equipo	X	X	X	X	X	X	O
Id_Equipo_Salida	-	-	-	-	-	X	-
Canal_Entrada	-	X	X	X	-	-	O
Troncal	-	X	X	X	-	-	O
Modem_DS0	-	X	X	X	-	X	O
Paquetes_de_Host	X	-	-	-	-	-	O
Paquetes_a_Host	X	-	-	-	-	-	O
Id_Host/Ruta	X	-	-	-	-	-	O
Flags_Finalización_1	X	-	-	-	-	-	O
Id_Gateway	X	-	-	-	-	-	O
Numero_de_Secuencia_Llamada	X	-	-	-	-	-	O
Causa_Diagnóstico	X	-	-	-	-	-	O
Dirección_Origen	X	-	-	-	-	-	O
Puerto_Origen	O	-	-	-	-	-	O
Dirección_Destino	X	-	-	-	-	-	O
Puerto_Destino	O	-	-	-	-	-	O
Indicador_Velocidad	-	X	X	X	-	-	O
Id_Origen	-	X	X	X	-	X	O
Id_Destino	-	X	X	X	X	X	O
Flags_Finalización_2	-	X	X	X	X	-	O
Id_Red	-	X	X	X	-	-	O
Id_Subred	-	X	X	X	-	-	O
Indicador_Marcación	-	-	X	-	-	-	O
Puerto	-	-	-	X	-	-	O
Canal_Salida	-	-	-	-	-	X	-
Flags_Finalización_3	-	-	-	-	-	X	-
Tipo_Fichero	X	X	X	X	X	X	X
Total campos	17	17	18	18	9	12	31

X: Disponible O: Opcional -: No Disponible

Tabla 2. Campos disponibles en información de Contabilidad

- Tipo 4: La información en este tipo de ficheros está estructurada en pares clave-valor, donde el valor está en hexadecimal y en ocasiones vendrá precedido del número de caracteres de que consta el valor. A la hora de llevar a cabo el pre-procesado deberá tenerse en cuenta la clave para transformar el valor adecuadamente (Numérico, BCD, Fecha, etc.) y saber si el campo en cuestión dispone de indicador de longitud.

Consolidación de la información

Según el tipo de fichero de origen, existen claves en la Tabla 2 que son mutuamente excluyentes y aportan una información equivalente. Puede ser el caso, por ejemplo, de *Id_Origen* y *Dirección_Origen*: Aun siendo de naturalezas diferentes (La primera generalmente un número de teléfono, y la segunda una dirección IP), en un registro dado sólo puede aparecer una de las dos y, en cualquier caso, ambas identifican el *origen* del registro. Existe coalescencia entre estas claves y es mejor unir ambas bajo una única sobre la que buscar. De este modo se simplifica la búsqueda al usuario de la herramienta, quien hará referencia a una clave *origen*, sin que le importe la naturaleza del fichero del que se extrajo la información (o de la propia información) y si tenía que buscar por *Id _Origen* o *Dirección_Origen*.

Esta consolidación se realiza en el momento de indexar la información, y puede optarse por eliminar o mantener (por completitud) los campos fusionados, según la magnitud del impacto en el rendimiento que ello tenga.

Adición de información

Existe cierta información que puede inferirse a partir de la disponible para cada registro. Por este motivo, en esencia, es redundante pues no añade entropía. Sin embargo dicha información puede resultar muy útil a la hora de realizar búsquedas. Por ejemplo, si quisiéramos extraer información de contabilidad acerca del comportamiento de un *cliente* concreto, no disponemos de ningún campo sobre el que realizar la búsqueda (Cada *cliente* se encuentra relacionado con una serie de *Id_Destino* que tiene asignados). Se pueden plantear varias alternativas para afrontar esta necesidad:

a) Hacer la búsqueda por todos los *Id_Destino* que pertenecen al *cliente*, pero eso obligaría o bien a que el usuario que la realiza tenga el conocimiento de los mismos, o a desarrollar una interfaz intermedia que hiciera este trabajo.

b) Computar el *cliente* en el momento de la indexación y cargarlo con el resto de los datos. De este modo estaría disponible a la hora de realizar las búsquedas, pero a costa de aumentar significativamente el tamaño de los índices con cada nuevo campo añadido a los datos.

c) Computar el cliente en el momento de la búsqueda, que supone una solución de compromiso entre a) y b): Permite la búsqueda por cliente (como en b), sin

aumentar el tamaño de los índices, a costa de aumentar el tiempo de respuesta. Splunk permite la adición de información de forma automática para que esté disponible para la búsqueda de forma transparente para el usuario de la herramienta (Ver el apartado Rendimiento con adición de información en tiempo de búsqueda en la página 47).

Cabe señalar que optar por la tercera opción, no implica un impacto negativo permanente en todas las búsquedas, pues la adición de información automática puede configurarse para que esté activa sólo bajo ciertas circunstancias, de modo que no cargue el sistema cuando no sea necesario. Incluso si no se tomase esta precaución, la inteligencia introducida en Splunk evitará, generalmente, ese impacto negativo en el rendimiento de las búsquedas (de nuevo remitirse al apartado Rendimiento con adición de información en tiempo de búsqueda).

Visualizaciones

Splunk ofrece un amplio abanico de visualizaciones de datos, desde valores individuales o tablas, pasando por gráficas de líneas, barras o tarta, hasta gráficas más complejas, como de burbujas o basadas en geolocalización.

Todas estas visualizaciones pueden agruparse en *dashboards* (o paneles) para poder tener acceso a una gran cantidad de información de un solo vistazo, lo que facilita enormemente la monitorización de la red. Además, como ya se adelantó y se verá en el apartado Visualización y Desglose: Drill-down, la funcionalidad de *drill-down* ofrecida por este producto resulta de enorme ayuda para, una vez localizada una anomalía, poder recuperar la información relacionada con ella de forma inmediata.

Puede disponerse de tantos paneles como se desee, y pueden asignarse por perfiles de usuarios, de modo que cada departamento de la compañía tenga acceso a la información más relevante ajustada a su función.

Alarmas

Como complemento a la monitorización de la red mediante los diferentes paneles de visualización que se configuren, ya se mencionó la gran utilidad de que el propio sistema sea capaz de generar avisos de problemas que puedan surgir.

A tal fin Splunk cuenta con un versátil apartado de Alarmas para responder ante eventos concretos. En esencia consiste en la ejecución de búsquedas, bien en tiempo real, bien de forma periódica, y abarcando una ventana de duración configurable.
Según los resultados obtenidos en la búsqueda, puede desencadenarse (o no) una única acción o una acción por cada resultado. Para evitar que ante un problema de larga duración (una caída de un circuito, o fallo de un servidor, por ejemplo), se generen continuamente acciones, Splunk ofrece la opción de, tras ejecutar una acción, deshabilitar la generación de nuevas durante un período de tiempo determinado. Las acciones desencadenadas pueden ir desde registrar un evento dentro de un índice de Splunk hasta la ejecución de un script, pasando por el envío de un correo o un POST HTTP a una URL dada.

Implementación

Configuración de la arquitectura de pruebas

Para la realización del estudio del funcionamiento de la arquitectura propuesta se ha utilizado la versión de evaluación de Splunk v.6.6.1, que dispone de la práctica totalidad de las funcionalidades del producto con la limitación de una indexación diaria máxima de 500MB y dos meses de período de pruebas.

Como hardware se ha utilizado un ordenador con procesador Intel Core I7-6700HQ @ 2.60GHz y 64 GB de RAM, con sistema operativo Windows10. Sobre esta plataforma se definieron 7 máquinas virtuales (VMWare Workstation 12) con 40GB de disco duro y 2GB de RAM reservadas cada una, con un sistema operativo Ubuntu 16.04.

Cada una de las máquinas fue configurada tal y como se muestra en la Figura 13, conformando una arquitectura básica sobre la que poder realizar pruebas.
El clúster configurado consta de un Factor de Replicación 3 y Factor de Búsqueda 2, lo que significa que en todo momento los nodos Peer mantienen 3 copias de los datos enviados por los nodos alimentadores, y de ellas, dos cuentan con las estructuras de índices necesarias para poder realizar búsquedas sobre ellos. De este modo, si se perdiese uno de los nodos Peer, las búsquedas podrían continuar realizándose normalmente, y aún se dispondría de otra copia adicional de los datos como seguridad adicional.

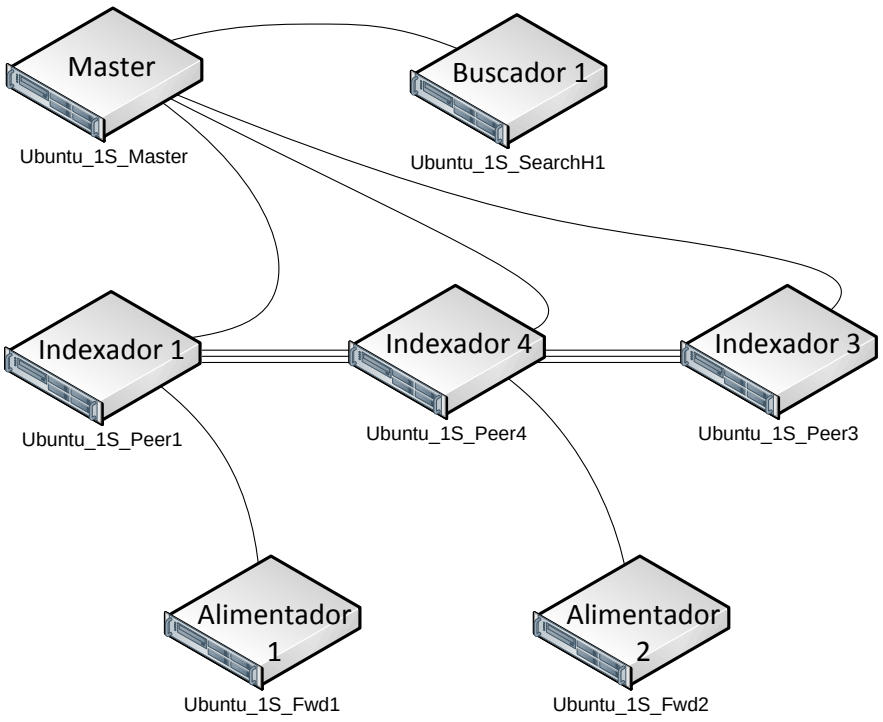


Figura 13. Arquitectura de evaluación

La instalación por defecto de Splunk indexa una importante cantidad de información sobre el estado del sistema periódicamente. Para evitar que dicha información se mezclase con aquella empleada en este estudio, se definió un *índice* propio.
En esencia, el sistema de almacenamiento de Splunk separa los índices en diferentes carpetas, de modo que al crear un índice que contenga nuestros datos, aseguraremos que en cada momento contenga únicamente la información que hayamos inyectado voluntariamente en el sistema, esencial para las medidas de rendimiento que se realicen no se vean "contaminadas".

Nótese que los resultados obtenidos no se pueden considerar desde el punto de vista absoluto, pues los recursos de las máquinas virtuales (procesadores) son compartidos. Sin embargo sí permiten una visión relativizada del impacto de ciertas prácticas o escenarios.

Inyección de datos

Para este trabajo se han cargado datos de contabilidad de dos naturalezas (Los tipos 2 y 4 descritos en el apartado *Información disponible* en la página 38) a lo largo de una semana.

En los nodos alimentadores se ha partido de los datos originales, y se han transformado a un formato clave=valor en texto claro, volcando la salida en ficheros monitorizados por las instancias Splunk Forwarder, que las han enviado a los nodos Peer.

El script que se ha elaborado para la transformación de los datos originales a un formato clave=valor, teniendo en cuenta las singularidades presentadas en el apartado Información disponible, puede verse en el Anexo 1. Script clave-valor.

Cuando se describieron los Indexadores de Splunk, se adelantó que la documentación de este producto estima que para la información proveniente de un *syslog* los datos tienen una compresión aproximada del 85% y que los índices pueden suponer un 35% del espacio que ocupa en disco la información original. En nuestro caso concreto, estos fueron los volúmenes original y almacenado:

	Total (kb)
Forwarder1 tnc	319.743
Forwarder1 tlv	547.409
Forwarder2 tnc	6.349.800
Forwarder3 tlv	182.528
Total	7.399.481

	Total (kb)	Datos (kb)	Indices (kb)	Réplica	Búsqueda
Peer5	2.918.704	1.298.832	1.619.872	Si	Si
Peer6	2.916.176	1.298.840	1.617.336	Si	Si
Peer7	1.296.092	1.296.036	56	Si	No
Total	5.834.880	2.597.672	3.237.208		
Compresión	21,1%	64,9%	56,3%		

Tabla 3. Espacio ocupado por datos e índices en Splunk

Puede observarse que en el caso de los registros de contabilidad de la compañía objeto de estudio, los resultados son peores- menor ratio de compresión- que los previstos para un *syslog*.

En el caso de los datos, se han comprimido un 65% en lugar del 85%. El motivo radica en que *syslog* generalmente produce registros con un importante contenido textual, cuyo conjunto de mensajes y palabras es bastante acotado, lo que permite un elevado ratio de compresión, mientras que los registros de contabilidad resultan más aleatorios en la naturaleza de su composición y los valores de sus campos.

En el caso de los índices, no se puede ser concluyente a la hora de afirmar porqué el ratio de compresión es un 10% menor con los datos de contabilidad que el previsto por Splunk sin contar con más información de los datos que utilizaron como referencia para estas cifras. No se ha encontrado un promedio de tamaño de un mensaje *syslog* definitivo, pues depende de la naturaleza del sistema que lo genera. Se ha tomado como referencia el empleado por Cisco para el dimensionamiento de servidores de syslog[xi], 244 bytes. En nuestro caso, los ficheros de registro tnc, responsables de la mayoría de la información almacenada en Splunk, los mensajes tienen un tamaño promedio de 157 bytes, mucho menor. Esto puede explicar el mayor tamaño de los índices, pues a igual volumen de información, existen más registros a indexar.

Cumplimiento del Teorema CAP

Como es de esperar, Splunk se encuentra limitado por el Teorema CAP. Mientras el nodo Maestro está activo, prima la consistencia de los datos frente a la disponibilidad, lo que consigue manteniendo un listado de cubos primarios. Recordemos que un cubo es la manera que Splunk almacena la información: En esencia se trata de directorios dentro de los nodos Indexadores. Dado que la misma información – cubos - se encontrará replicada en varios indexadores para aumentar la disponibilidad de la misma, el nodo Maestro etiqueta una de las réplicas de cada cubo como primaria dentro del listado completo que mantiene. Cuando un nodo Buscador solicita la ubicación de los cubos sobre los que realizar una consulta, el Maestro le proporcionará la de los cubos primarios.

Si un nodo Indexador estuviese caído y contuviese cubos primarios, el Buscador no podría acceder a él y avisaría de ello al usuario ante el riesgo de devolver información incompleta, sin arriesgarse a devolver información inconsistente acudiendo a otro Indexador.

¿Cuándo detecta el nodo Maestro la caída de un nodo Indexador? Cada nodo indexador manda un mensaje de latido con período variable (1 segundo por defecto). Si el nodo Maestro no recibe latidos de un Indexador por un período determinado (60 segundos por defecto), lo marca como no disponible, re-etiqueta como cubos primarios los de otros nodos disponibles, y comienza, si procede, los mecanismos necesarios para recuperar los factores de búsqueda y réplica.

El comportamiento se ha verificado cortando la comunicación de uno de los nodos Indexadores (*Peer5*): Cuando se realizó una consulta desde el Buscador antes de que el Maestro marcase el nodo como caído, devolvió el siguiente mensaje:

Figura 14. Advertencia ante consulta con Indexador caído no detectado por el nodo Maestro

Transcurrido un minuto, el Maestro detectó la caída del nodo Indexador y lo marcó como tal, pudiendo entonces completarse normalmente las consultas desde el Buscador, una vez el Maestro hubo marcado otros cubos disponibles como Primarios:

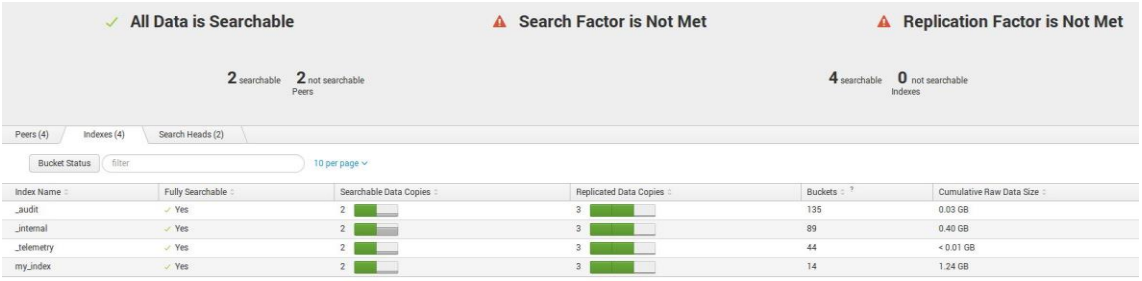


Figura 15. Nodo Maestro detecta caída de Indexador con cubos Primarios

Se probó a disminuir el Timeout del latido a 30 segundos y, como era de esperar, el Maestro redujo a este tiempo la detección de la caída de un nodo. Sin embargo, hay que entender las consecuencias de reducir este valor, pues aunque se acorta el período de indisponibilidad, ante un parpadeo de las comunicaciones de un Indexador podría comenzarse a replicar índices y datos de cara a reestablecer los factores de búsqueda y réplica.

Cuando el nodo caído reestablece las comunicaciones, lo indica inmediatamente al Maestro con su siguiente latido. No obstante el Maestro no lo marca como activo hasta que sus índices y datos se han sincronizado con el resto de nodos activos. Si la caída ha sido breve, se ha comprobado que esto es cuestión de unos pocos segundos, llevando más tiempo cuando mayor es el volumen de información a sincronizar, y dependiendo del ancho de banda disponible en la red.

Por otro lado, el comportamiento de los Reenviadores ante la caída de un Indexador varía considerable dependiendo de la configuración de los mismos. En la arquitectura desplegada para este estudio cada Reenviador estaba configurado para mandar la información a un único Indexador (con el fin de poder determinar fidedignamente donde sería indexada). Cuando, al caer un Indexador, el Reenviador correspondiente no tuvo

dónde mandar la información, la almacenó en memoria[17] hasta que el Indexador volvió a conectarse, momento en el cual se envió toda la información atrasada. Pudo comprobarse, además, que los *timestamps* de los eventos fueron los correctos, creados por el Reenviador en el momento de recibir la información, y no generados por el Indexador cuando recibió la información.

Los Reenviadores también pueden configurarse para realizar balanceo de carga entre varios Indexadores. En este caso, manda información a un Indexador durante un intervalo configurable de tiempo, transcurrido el cual escoge un nuevo Indexador de forma aleatoria (verificando previamente que pueda alcanzarlo), volviendo a comenzar el proceso. Los Indexadores entre los que repartir la información pueden estar definidos de forma estática en el Reenviador, o éste puede configurarse para consultar al nodo Maestro para que le proporcione el listado. En este caso, además, existe la opción de que el reparto de carga se realice de forma proporcional al tamaño de almacenamiento total (no el disponible) de cada Indexador. Si un Indexador falla teniendo el balanceo de carga activo, el Reenviador se limita a mandar la información a otro Reenviador disponible tan pronto detecta el problema.

Figura 16. Buffering de Reenviador ante caída de Indexador

[17] Para el control de la información enviada al Indexador, el Reenviador divide el flujo de datos en bloques de unos 64kB, almacenándolos en memoria hasta recibir el acuse de cada uno de ellos desde el Indexador, una vez este ha procesado y almacenado los datos (y la información de índice). El tamaño de la cola de espera es configurable, para poder ajustarse al volumen de información transmitida.
Se ha podido comprobar, sin embargo, que si los datos de origen permanecen en el Reenviador (pe. Por estar leyéndolos de un fichero en lugar de como un flujo de datos), éste no está sujeto a la limitación en el tamaño de la cola: Se inyectaron 504MB de datos sobre un fichero monitorizado por el Reenviador, estando el Indexador caído, y cuando éste se reestableció, pudo indexar la totalidad de los mismos. Sin embargo, en el mismo escenario con la salvedad de que el fichero monitorizado se borró antes de recuperarse el Indexador, sólo se indexó 1MB.

En el caso de que sea el nodo Maestro el que falle, el sistema puede seguir funcionando con normalidad, utilizando la última información recibida desde el mismo antes de fallar. Solo a largo plazo o si falla algún otro nodo, podrán surgir problemas, al no actualizarse la información de cubos primarios o no poder dar comienzo a los mecanismos de recuperación de los factores de búsqueda y réplica.

Rendimiento con adición de información en tiempo de búsqueda

En el apartado *Adición de información* se expuso la funcionalidad de Splunk de incorporar información a los resultados en tiempo de búsqueda. Splunk lleva a cabo esta tarea mediante *lookups*[18] automáticas sobre tablas que cruza con los datos extraídos en cada búsqueda. Esta característica permite al usuario realizar consultas sobre campos que no existen en el conjunto de datos originales pero que pueden inferirse a través de ellos.

Por ejemplo, si tomamos como punto de partida los registros de contabilidad disponibles, los equipos que han generado cada uno de ellos aparecen identificados por un código numérico. Si disponemos de una tabla con las correspondencias entre dichos códigos numéricos y los nombres de los equipos, podemos habilitar una *lookup* automática para poder realizar búsquedas utilizando los nombres de los equipos en lugar de sus contrapartidas numéricas. Nótese que estas *lookups* se aplican en tiempo de búsqueda, de modo que no tienen efecto alguno sobre el tamaño de los datos almacenados.

Lógicamente, esta funcionalidad tiene una contrapartida, que es el tiempo adicional requerido para incorporar la información adicional a los datos almacenados a la hora de realizar la búsqueda. Sin embargo ventajas tales como el ahorro en espacio de almacenamiento y la posibilidad de poder actualizar o añadir esta información a posteriori, bien pueden compensar la espera.

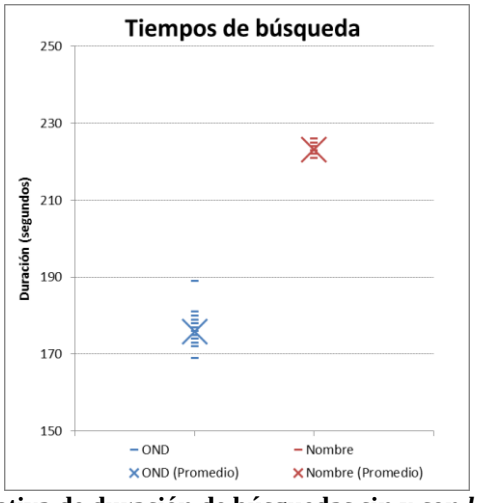


Figura 17. Comparativa de duración de búsquedas sin y con *lookups* automáticas
Datos medidos para un total de 43.997.159 registros y una tabla de *lookup* de 1458 elementos

[18] Búsquedas, pero se prefiere el término anglosajón aquí para diferenciar estas búsquedas dentro de tablas, de aquellas realizadas por el cliente sobre el conjunto de datos.

La Figura 17, muestra el tiempo de una búsqueda que devuelve el número de operaciones que se han realizado por cada equipo de la red, listándose bien por el identificador numérico de cada equipo (OND) o bien su nombre (obtenido mediante un *lookup* automático). El impacto es significativo, pues supone un incremento de 50s sobre una búsqueda de 175s, es decir un 29% más.

Esto significa que se debe tener muy presente el impacto de las *lookups* automáticas pues pueden lastrar considerablemente el rendimiento de las búsquedas. ¿O no?

Optimización automática de Splunk

Afortunadamente, Splunk cuenta con un nivel aceptable de inteligencia y es capaz de optimizar en cierto grado las búsquedas introducidas para mejorar el rendimiento y, en última instancia, el tiempo en completarlas. En el ejemplo anterior, el efecto de habilitar una *lookup* automática para listar los equipos por nombre en lugar de por el identificador numérico presente por defecto en cada registro de contabilidad, suponía un incremento de tiempo de casi el 30%. Se trata de un aumento considerable, debido a que dicha *lookup* añade a cada registro el nombre del equipo que lo generó, en función de su identificador numérico. Sin embargo, si dejamos habilitada la *lookup* automática, pero decidimos ignorar los nombres de los equipos y listar por sus identificadores numéricos, estos son los resultados:

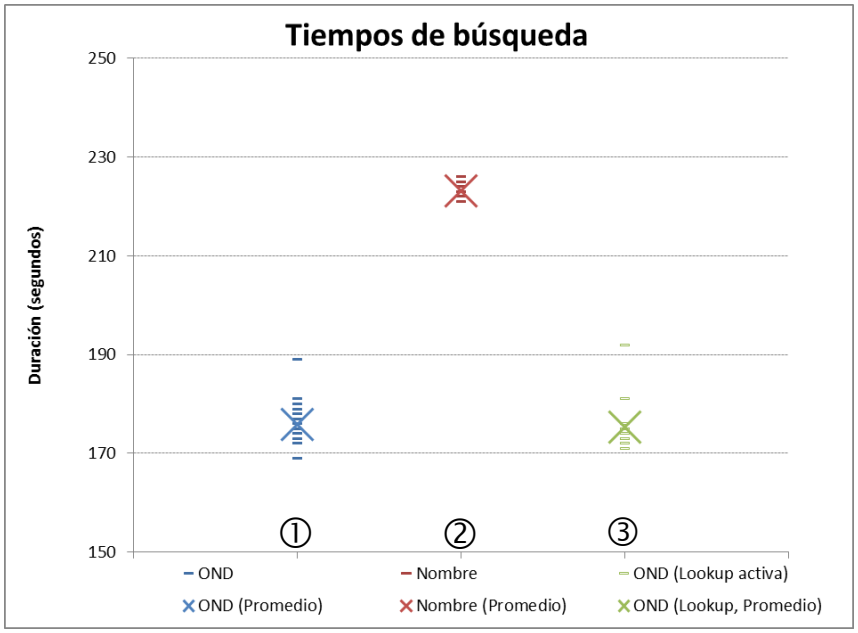


Figura 18. Comparativa de duración de búsquedas con *lookups* automáticas, empleándolas o no
Datos medidos para un total de 43.997.159 registros y una tabla de *lookup* de 1458 elementos

Se observa que las búsquedas por OND tienen la misma duración, sin importar que la *lookup* estuviese activa (③) o no (①). El motivo es que Splunk analiza la búsqueda solicitada y lleva a cabo internamente las modificaciones necesarias para optimizar su rendimiento. En este caso, opta por no realizar la *lookup* en (③) dado que la búsqueda introducida no requiere del nombre de los equipos para llevarse a cabo. Al consultar una de las herramientas de administración de Splunk, se puede comprobar que, efectivamente, sólo ② dedica tiempo a *lookups*:

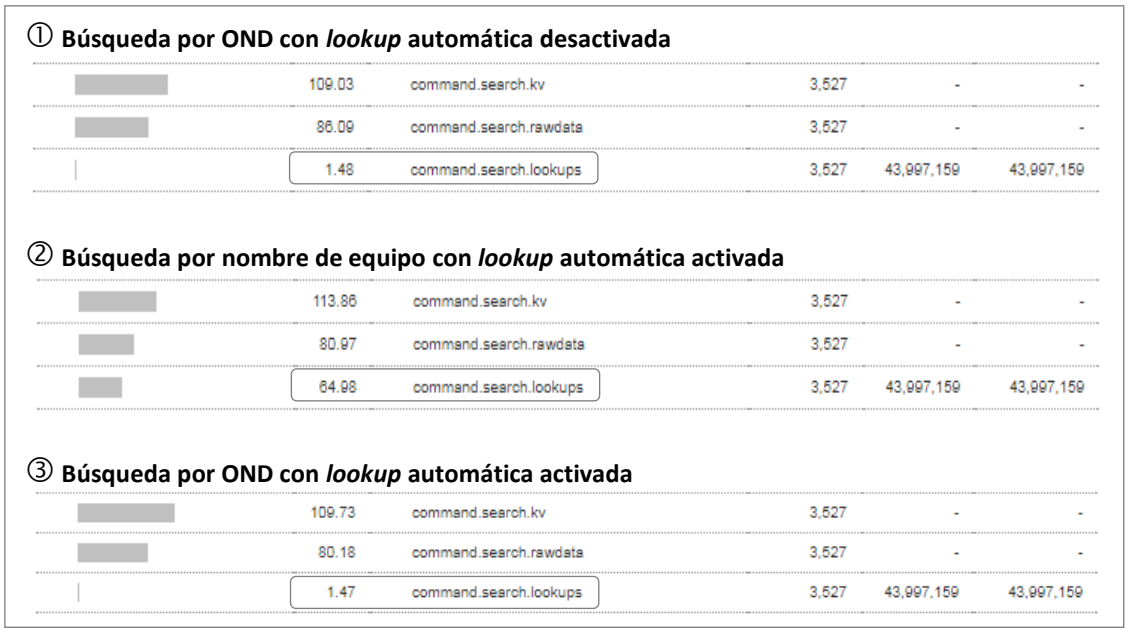


Figura 19. Tiempos empleados en *lookups* al ejecutar búsquedas
Datos medidos para un total de 43.997.159 registros y una tabla de *lookup* de 1458 elementos

La optimización de Splunk no se limita a eliminar elementos de la búsqueda que no resultan necesarios. En la normalización de la búsqueda (como se define en Splunk), también presta atención a otros aspectos, como puede ser el orden en que se escribieron las diferentes operaciones de que consta la misma.
Tomemos, por ejemplo, el caso en que queramos contar todas las operaciones que atraviesan un nodo concreto de la red de comunicaciones, sabiendo que el nombre de cada equipo siempre comienza por tres letras que identifican el nodo en que se encuentra instalado. Además, también sabemos el país en que se encuentra dicho nodo, y esta información también se encuentra presente en cada registro de contabilidad, de modo que filtraremos por dicha clave, para reducir en un primer momento el número de registros sobre los que aplicaremos la *lookup* para obtener el nombre del equipo que generó cada uno.

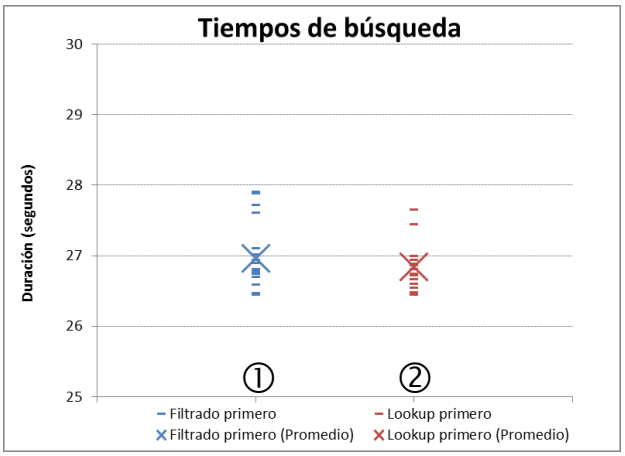


Figura 20. Tiempos empleados según orden en búsqueda
Datos medidos para un total de 43.997.159 registros y una tabla de *lookup* de 1458 elementos

En la figura anterior, ① representa el caso expuesto: Filtrado previo por identificador de país y entonces *lookup* para obtener el nombre de cada equipo y, finalmente, filtrado por nombre de equipo/nodo.
El caso ②, debiera haber sido subóptimo, pues primero se hace la *lookup* para obtener los nombres de los equipos de todas las operaciones y luego se filtra por nombre de equipo/nodo y finalmente por país (innecesario). Sin embargo, ambas búsquedas presentan funcionamientos idénticos. Acudiendo nuevamente a la herramienta de administración de tareas de Splunk mencionada anteriormente, vemos el motivo:

Figura 21. Búsquedas normalizadas por Splunk

Pese a que las búsquedas que hemos introducido son diferentes, Splunk internamente las ha reescrito para mejorar el rendimiento de las mismas, dando como resultado final dos búsquedas iguales, que explican la idéntica duración de ambas.

Consultas y valores experimentales

A continuación se muestran las consultas reales y los resultados obtenidos para la elaboración de este apartado. Nótese que los resultados de la Figura 18 incluyen a los de la Figura 17:

Recuento de operaciones por OND. Lookup automática desactivada
```
search index=my_index | stats count by OND
```

Recuento de operaciones por nombre de equipo. Lookup automática activada
```
search index=my_index | stats count by DEVICENAME
```

Recuento de operaciones por OND. Lookup automática activada
```
search index=my_index | stats count by OND
```

La siguiente tabla muestra los tiempos empleados en realizar las búsquedas de la Figura 18, sobre una base de 43.997.159 registros y una tabla de *lookup* de 1458 elementos:

		Promedio	Duraciones de búsquedas medidas (segundos)								
1	Recuento por OND. Lookup automática desactivada	**176,9**	173	181	192	174	176	174	174	173	175
			174	175	175	171	172	173	175	176	173
2	Recuento por nombre de equipo. Lookup automática activada	**223,8**	225	222	226	222	223	222	226	224	224
			222	222	223	225	221	224	222	222	223
3	Recuento por OND. Lookup automática activada	**175,9**	173	189	175	179	169	181	172	178	174
			169	172	175	175	174	176	177	180	178

Tabla 4. Comparativa de duración de búsquedas con lookups automáticas, empleándolas o no

Igualmente, aquí se presentan las búsquedas empleadas y datos obtenidos para la representación de la Figura 20:

Recuento de operaciones a través de un nodo de red. Filtrado de país primero

```
search index=my_index NID=34 | lookup luNACID NACID_NAC_ID AS OND output NACID_DEVICENAME | search NACID_DEVICENAME=mad* | stats count
```

Recuento de operaciones a través de un nodo de red. Lookup de equipo primero

```
search index=my_index | lookup luNACID NACID_NAC_ID AS OND output NACID_DEVICENAME | search NACID_DEVICENAME=mad* | search NID=34 | stats count
```

		Promedio	Duraciones de búsquedas medidas (segundos)								
1	Filtrado por código de país primero	**26,9**	26,5	27,0	27,9	26,9	26,8	26,6	26,5	26,5	26,8
			26,9	26,5	26,7	27,1	26,8	26,7	27,7	27,6	27,9
2	Lookup de nombre de equipo primero	**26,8**	26,9	26,5	26,7	26,6	26,7	27,0	26,6	26,8	26,9
			26,5	26,7	27,5	26,9	26,9	26,8	27,7	26,9	26,8

Tabla 5. Tiempos empleados según orden en búsqueda
Datos medidos para un total de 43.997.159 registros y una tabla de *lookup* de 1458 elementos

Rendimiento de búsquedas. La importancia de un código óptimo.

Splunk optimiza las búsquedas introducidas pero su capacidad para llevarlo a cabo se ve mermada conforme aumenta la complejidad de las mismas. Por este motivo es de suma importancia diseñar las búsquedas teniendo en cuenta los resultados finales que desean obtenerse. Desafortunadamente no siempre el camino más rápido es el más simple.

En el siguiente ejemplo se plantea una búsqueda que muestre las operaciones que se han cursado a través de grupos de circuitos. La información de los circuitos que pertenecen a cada grupo se encuentra en una tabla sobre la que se realizará una operación de *lookup*. El planteamiento más simple consistiría en fijar un *lookup* automático de modo que se dispusiera del nombre de cada grupo de circuitos a la hora de realizar una búsqueda ①. Si sólo estuviéramos interesados en un único grupo, bastaría entonces con filtrar por el nombre de dicho grupo ③. Sin embargo, otra opción sería buscar primero en los datos aquellas operaciones para los circuitos en cuyos grupos estamos interesados y después cruzar estos resultados con la tabla para obtener el nombre de dichos grupos ②④.

Como se puede apreciar en la Figura 22, si nuestro objetivo es el listado completo de los grupos de circuitos con el número de operaciones para cada uno, el rendimiento de la opción *lookup* ① supera considerablemente al de filtrado previo ②, lo que, por otro lado, es obvio dado que en realidad la opción ② estaría consumiendo recursos sin llegar a filtrar nada en realidad.

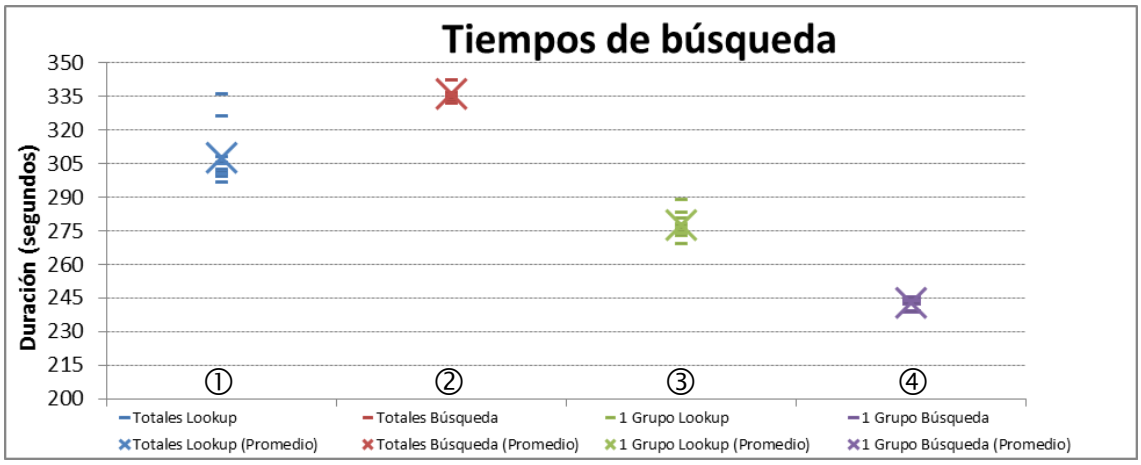


Figura 22. Comparativa de búsquedas según código utilizado
Datos medidos para un total de 43.997.159 registros y una tabla de Grupos con 490 elementos

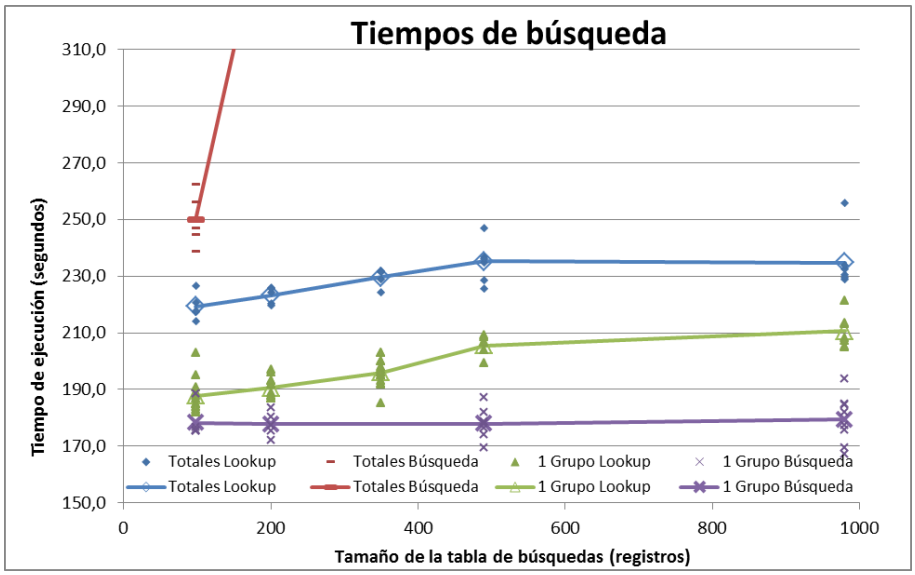
Sin embargo, si nuestra intención es obtener el número de operaciones a través de un único grupo, el rendimiento de búsqueda mediante un filtrado previo ④ es superior al de la opción de *lookup* y filtrado posterior ③. Aunque en ambos casos Splunk recorrerá todos los datos sobre el período de tiempo requerido, le resulta más costoso añadirles a todos una información extra y luego filtrar por dicha información, en lugar de recuperar tan sólo aquellos que resultan de interés. Este comportamiento tiende a invertirse cuanto mayor es el número de parámetros de búsqueda (en este caso el número de circuitos de interés), hasta llegar al caso extremo de ②.

Rendimiento en función del volumen de información.

Como se ha visto en el apartado anterior, el diseño de las operaciones sobre los datos resulta de suma importancia para optimizar el tiempo necesario para completarlas. Si a esto se le suma el aumento de dicho tiempo inherente al incremento de la cantidad de información que debe procesarse, el citado diseño puede llegar a hacerse crítico.

Volviendo al ejemplo del apartado anterior, la Figura 23 muestra los distintos tiempos empleados en completar cada operación conforme varia el tamaño de la tabla de búsquedas que contiene la relación de circuitos y grupo al que pertenecen. De los resultados obtenidos se aprecia la idoneidad del método basado en búsqueda previa de los circuitos de interés, en especial conforme aumenta el tamaño de la tabla de búsquedas, cuando el interés se centra en un único grupo de circuitos ④: Se realizará una búsqueda sobre dicha tabla para obtener los, pocos, circuitos de interés y a continuación se extraerán las operaciones correspondientes a dichos circuitos del conjunto de datos completo.

Si por el contrario interesa mostrar la información de todos los grupos de circuitos, el método por búsqueda ②, como ya se vio en el apartado anterior, muestra un rendimiento pésimo, ahora enormemente lastrado, además, con el aumento del tamaño de la tabla de búsquedas. En este caso interesa, de nuevo, la elección de un método basado en *lookup* ①, para el que Splunk se encuentra optimizado.



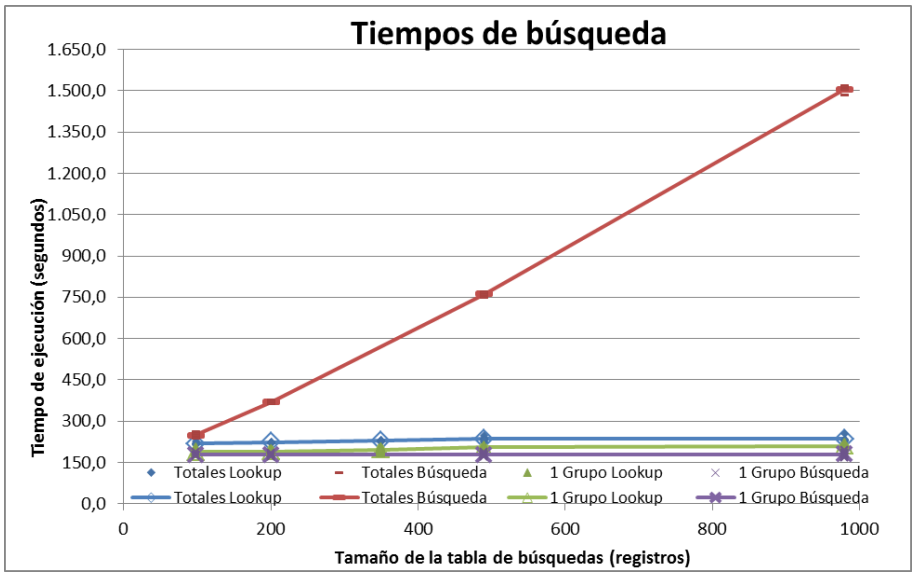


Figura 23. Comparativa de búsquedas según código utilizado y volumen Datos medidos para un total de 43.997.159 registros

Consultas y valores experimentales

A continuación se presenta el código de cada una de las consultas empleadas tanto en esta sección como en la previa y los resultados prácticos obtenidos para ellas:

Totales Lookup

```
index=my_index | lookup luNAC_Groups DG_OND AS OND, DG_OTN AS OTN OUTPUT DG_DEVICEGROUP
AS DEVICEGROUP | eval DEVICEGROUP=if(isnull(DEVICEGROUP),"unknown",DEVICEGROUP) | stats
count by DEVICEGROUP
```

Totales Búsqueda

```
index=my_index | search [|inputlookup luNAC_Groups | return 1000 OND=DG_OND, OTN=DG_OTN]
| lookup luNAC_Groups DG_OND AS OND, DG_OTN AS OTN OUTPUT DG_DEVICEGROUP AS DEVICEGROUP
| stats count by DEVICEGROUP
```

1 Grupo Lookup

```
index=my_index | lookup luNAC_Groups DG_OND AS OND, DG_OTN AS OTN OUTPUT DG_DEVICEGROUP
AS DEVICEGROUP | search  DEVICEGROUP="Jazztel_GN" | stats count by DEVICEGROUP
```

1 Grupo Búsqueda

```
index=my_index | search [|inputlookup luNAC_Groups | search DG_DEVICEGROUP="Jazztel_GN"
| return 1000 OND=DG_OND, OTN=DG_OTN] | lookup luNAC_Groups DG_OND AS OND, DG_OTN AS OTN
OUTPUT DG_DEVICEGROUP AS DEVICEGROUP | stats count by DEVICEGROUP
```

Para el estudio del impacto de la variación del tamaño de la tabla de búsquedas empleada para establecer la correspondencia entre circuitos y grupos de circuitos se emplearon diferentes versiones de la tabla *luNAC_Groups* de 98, 200, 350, 490 y 980 elementos.

Las gráficas se elaboraron a partir de valores promedios de varias iteraciones sobre el mismo caso, en un intento de minimizar el efecto que otros procesos que pudieran estar ejecutándose en la máquina en ese momento pudieran introducir. A continuación se muestran los datos correspondientes a:

- Figura 22. Comparativa de búsquedas según código utilizado

		Promedio	Duraciones de búsquedas medidas (segundos)										
1	Totales Lookup	**307,4**	326	305	301	301	300	299	336	297	308	305	302
2	Totales Búsqueda	**336,2**	343	332	335	337	333	336	334	337	342	334	
3	1 Grupo Lookup	**277,6**	283	278	269	281	273	276	273	289	278	275	
4	1 Grupo Búsqueda	**242,8**	245	239	245	239	243	246	243	243			

Tabla 6. Comparativa de búsquedas según código utilizado
Datos medidos para un total de 43.997.159 registros y una tabla de Grupos con 490 elementos

- Figura 23. Comparativa de búsquedas según código utilizado y volumen

	Elementos	Promedio	Duraciones de búsquedas medidas (segundos)													
Totales Lookup	98	225,5	233	224	227	220	224									
	200	229,4	230	226	232	226	232									
	350	235,8	238	238	235	238	230									
	490	234,8	237	235	229	226	247	236								
	980	235,2	229	231	256	233	230	234								
Totales Búsqueda	100	249,8	245	256	262	239	247									
	200	368,5	367	369	376	363										
	490	761,8	760	770	766	751										
	980	1.505,7	1.519	1.484	1.504	1.516										
1 Grupo Lookup	98	174,8	174	172	169	176	190	170	172	169	178	175	182	171		
	200	177,6	184	174	179	180	176	175	175	183	180	174	174			
	350	184,9	185	192	179	183	184	187	182	182	180	183	190	187	190	185
	490	205,5	209	207	199	204	208									
	980	210,5	221	213	208	205	214	205	207							
1 Grupo Búsqueda	100	178,3	175	175	176	188	177									
	200	177,8	175	180	172	184	177									
	490	178,0	178	177	182	179	170	187	174							
	1000	179,4	194	184	167	182	185	176	170	177						

Tabla 7. Comparativa de búsquedas según código utilizado y volumen
Datos medidos para un total de 43.997.159 registros

Visualización y Desglose: Drill-down

Splunk cuenta con una funcionalidad de gran utilidad para los operadores que gestionen las incidencias de la red y se trata del *desglose* o *drill-down*: Una gráfica puede mostrar una posible anomalía pero por su naturaleza aglutinadora y sintetizadora, enmascara la información necesaria para un diagnóstico. En lugar de realizar una búsqueda por dicha anomalía, la funcionalidad de *desglose* permite el acceso a la información relevante mediante un simple clic de ratón sobre la misma. La información que entonces se mostrará, a su vez es configurable.

La utilidad del *desglose* es tanto más útil cuanto más compleja es la visualización de la que procede y que, en ciertos casos, puede no ser siquiera reproducible. Aunque esta última afirmación puede parecer extraña, el siguiente ejemplo permitirá aclararla.

Tomemos como punto de partida un panel que muestre un diagrama de burbujas en el que los ejes X e Y representan los bytes recibidos y transmitidos en una operación, y el diámetro de las burbujas representan el número de operaciones con ese patrón de bytes. En operaciones como las de la red que nos ocupa, generalmente los patrones de bytes tienden a ser similares pues ante, por ejemplo, una venta con tarjeta de crédito, la información que viaja del datáfono hacia el centro autorizador será siempre la misma, los mismos campos, variando sólo el valor de los mismos (número de tarjeta de crédito, importe de la venta, código del comercio,…) y que, en cualquier caso, tendrán un tamaño prácticamente constante. Lo mismo ocurrirá con la respuesta del centro autorizador. De este modo, cada tipo de operación y respuesta tendrá un patrón de bytes intercambiados razonablemente homogéneo. Para aglutinar pequeñas variaciones en el

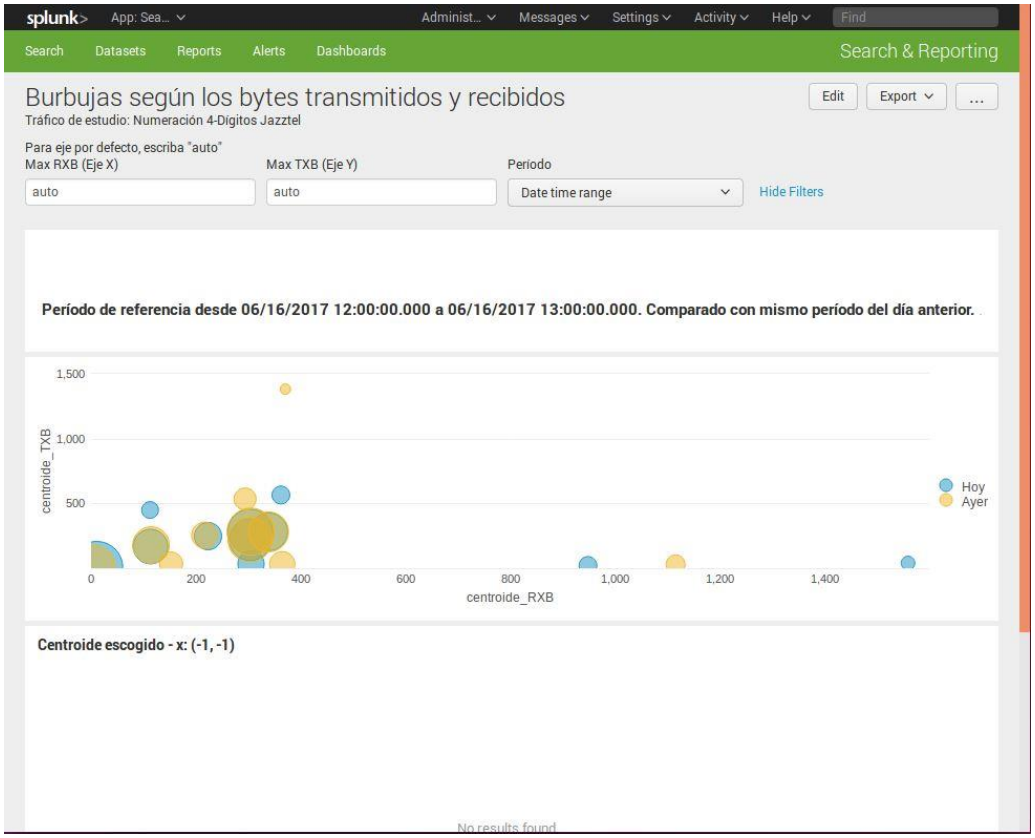


Figura 24. Panel con diagrama de burbujas de bytes transmitidos y recibidos

número de bytes y dado que a priori desconocemos los patrones de bytes estándar, podemos hacer uso de un algoritmo k-medias para llevar a cabo el agrupamiento. Es lógico pensar que los patrones de bytes en un momento dado del día se aproximen a los del día anterior a la misma hora[19]. Esto es lo que muestra la Figura 24.

En este panel, creado como ejemplo, se han incluido elementos que permiten al usuario escoger los valores máximos de los ejes X e Y, para poder obviar operaciones que hayan supuesto un error puntual, con valores de bytes anormalmente altos en cualquiera de los dos ejes (Figura 25). Así mismo, también puede escogerse el período de estudio. Adicionalmente, se ha habilitado un pequeño resumen con los datos de cada centroide accesible sólo con desplazar el cursor del ratón sobre aquel que se desee consultar (Figura 26):

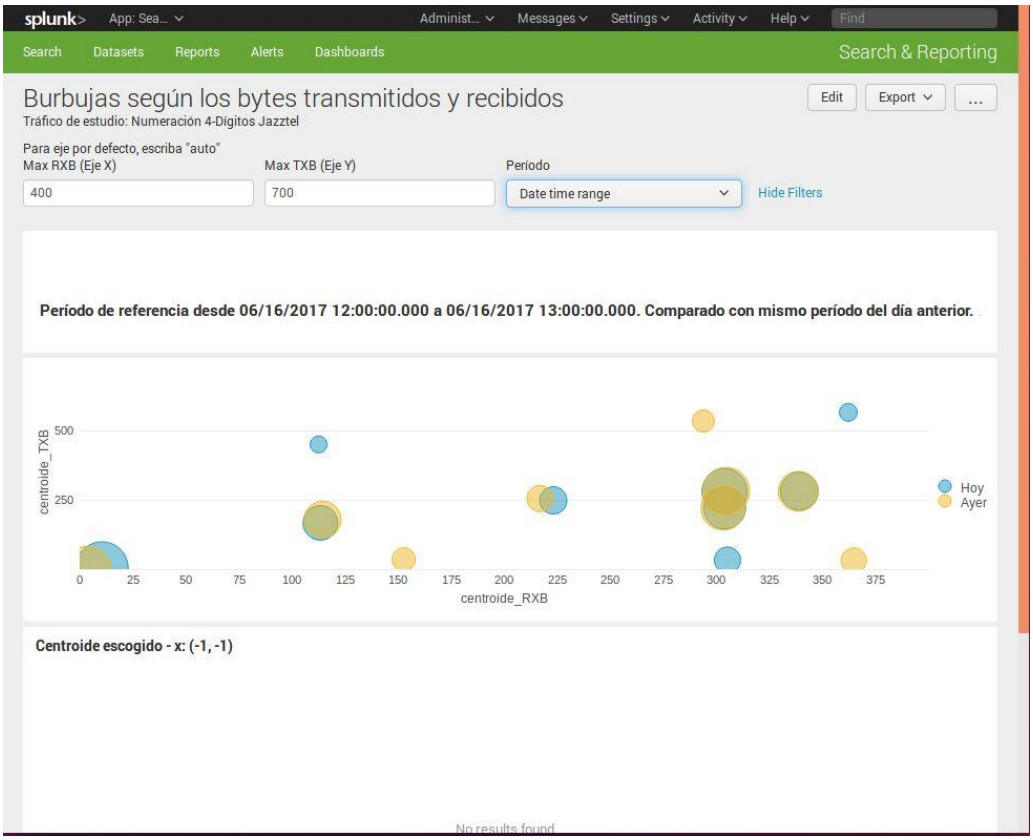


Figura 25. Panel con diagrama de burbujas de bytes transmitidos y recibidos. Tamaño de los ejes personalizado.

Figura 26. Información centroide

[19] El hecho de comparar los mismos horarios es importante, pues los tipos de las operaciones en un comercio suelen variar a lo largo del día (y con ello los patrones de bytes): Operaciones de apertura a primera hora de la mañana, ventas y devoluciones a lo largo del día, y cierres y consultas de totales a la hora del cierre.

El algoritmo k-medias parte de una distribución aleatoria inicial de los centroides que agruparán todos los puntos. Así, ante una posible anomalía representada por un centroide fuera de lugar con respecto al día anterior, el usuario no puede saber qué operaciones engloba dicho centroide, ni cómo buscarlas y es donde la funcionalidad de *desglose* de Splunk resulta de inestimable ayuda.

En este ejemplo se ha configurado el *desglose* de modo que al pulsar sobre un centroide se muestren todas las operaciones que engloba el mismo, escogiendo los campos más relevantes para realizar un diagnóstico, como hora, equipo (*lookup* automática), circuito, identificador de llamante, número llamado, bytes en cada sentido y código de finalización de la operación (Figura 27).

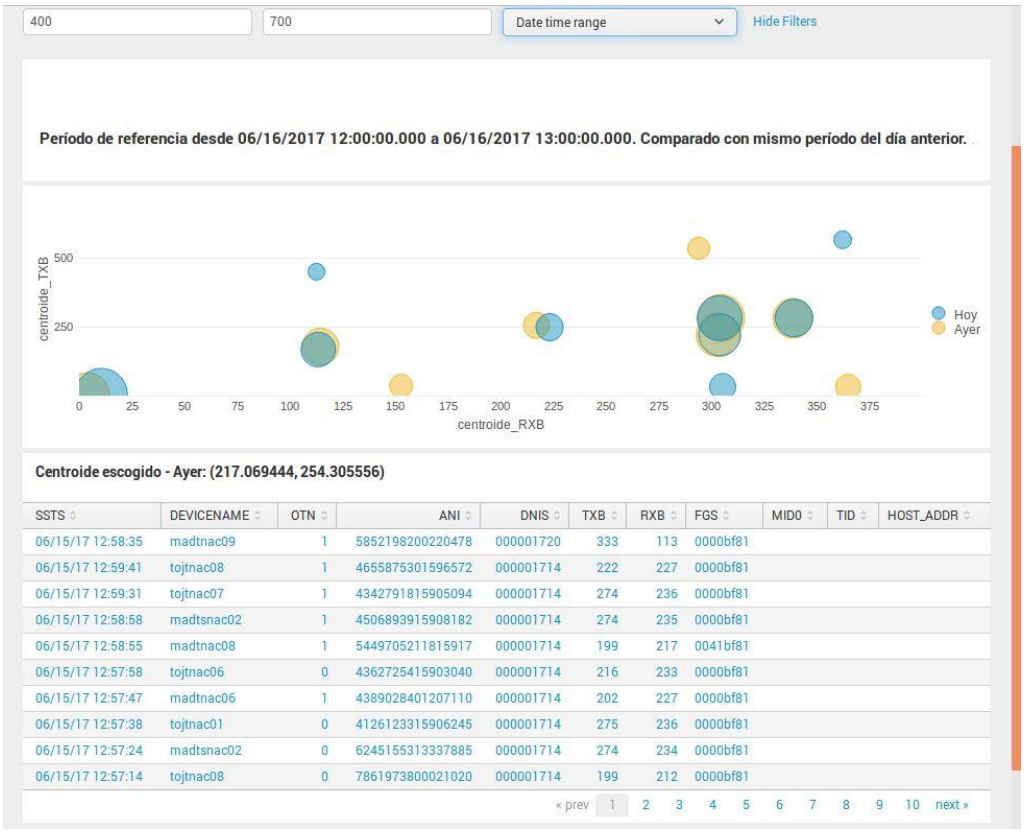


Figura 27. Ejemplo de desglose o drill-down.

Reutilización de búsquedas en paneles

Otra interesante funcionalidad ofrecida por Splunk es el poder reutilizar búsquedas dentro de los paneles, evitando tener que consultar innecesariamente el conjunto original de datos en varias ocasiones si la información requerida en varios puntos del mismo puede obtenerse a partir de la misma búsqueda.

Tomando de nuevo como ejemplo el panel con el diagrama de burbujas, los datos de dicho diagrama se obtienen a partir de una búsqueda s1 (consultar el Anexo 2. Panel con diagrama de burbujas). Dicha búsqueda se realiza al cargar el panel y cada vez que se

modifican los valores personalizables en el mismo (Valores máximos de los ejes e intervalo de interés), y sus resultados quedan almacenados temporalmente.
Para representar gráficamente el diagrama, se parte de dichos datos (referenciados mediante `search base="s1"`) y se calculan los porcentajes que representan sobre el total para mostrar dicha información cuando el cursor del ratón pase por cada centroide. De igual modo, cuando se pulsa sobre uno de los centroides, se vuelve a partir de la búsqueda s1 y se muestran únicamente las operaciones correspondientes a dicho centroide, filtrado esos resultados por las coordenadas del centroide escogido y el día al que pertenece, evitando así tener que consultar los datos originales (cosa que tampoco hubiera sido posible en este caso, pues no tendrían asociados los centroides asignados mediante el algoritmo k-medias).

Indexación de nuevos campos

Splunk permite buscar por cualquier campo presente en los datos, pero por defecto sólo indexa por la fecha y hora del registro, fuente, tipo de fuente y host, y es sólo para ellos que la búsqueda está optimizada.

Si bien desde Splunk se desaconseja, la herramienta permite indexar campos en, valga la redundancia, tiempo de indexación. Los motivos que alega son el impacto en:

- Tiempo de indexación: al añadir la carga adicional que supone indexar el nuevo campo)
- Tiempo de búsqueda: pues la adición del campo a los índices incrementa el tamaño de los mismos.

Se ha considerado interesante evaluar este impacto, de cara a poder tomar la decisión acerca de la idoneidad de indexar o no un determinado campo.

En primer lugar se indexó un nuevo conjunto de datos (8.182,73 MB, unos 45 millones de registros) donde el campo *DNIS*[20] (número/dirección llamado/a) se añadió con el alias *my_dnis* como nuevo campo en tiempo de indexación:

Conteo de número de operaciones de	Mejora	Tiempo (s)										
		Promedio	Pruebas									
Cada my_dnis	23,7%	149,3	178	167	148	146	149	137	144	139	151	134
Cada DNIS		195,7	194	214	203	193	231	186	182	188	184	181
my_dnis más común	10,1%	19,2	19,3	19,4	19,1	19,1	19,0	19,3	19,2	19,2	19,2	19,4
DNIS más común		21,4	21,5	21,1	21,2	21,4	21,4	21,6	21,1	21,2	21,9	21,2
my_dnis menos común	11,4%	0,23	0,31	0,19	0,17	0,44	0,20	0,16	0,31	0,21	0,18	0,16
DNIS menos común		0,26	0,44	0,19	0,24	0,28	0,17	0,37	0,23	0,17	0,31	0,23

*: El DNIS/my_dnis más común aparece 1.807.130 veces, mientras que el menos común aparece sólo 1 vez.

Tabla 8. Efecto en tiempo de búsqueda de indexar un campo

[20] Este campo es muy utilizado en la empresa de comunicaciones pues es básico para determinar el cliente al que pertenece, identificar la aplicación para la que se utiliza o al proveedor de comunicaciones del que proviene (y categoría dentro del mismo), por ejemplo.

Por otro lado se procedió a indexar un mismo bloque de datos añadiendo y sin añadir el campo *my_dnis* en tiempo de indexación, para comparar el efecto en el espacio en disco ocupado que tiene la introducción del mismo. En este caso se realizó con una cantidad de datos muy inferior (83,22 MB, poco más de 1 millón de registros), pues no era necesario evaluar tiempos de búsqueda[21]:

	Espacio en disco (MB)	Aumento
Sin nuevos campos indexados	65,45	13,8%
Indexando my_dnis	74,50	

Tabla 9. Efecto en espacio en disco de indexar un campo

Finalmente se decidió medir el efecto negativo que tiene el añadir un nuevo campo indexado sobre el tiempo de búsqueda de campos no indexados: Al aumentar el tamaño de los índices y de los datos indexados, las búsquedas deben verse, necesariamente, ralentizadas:

Conteo de número de operaciones de	Penalización	Tiempo (s)										
		Promedio	Pruebas									
Cada my_dnis		4,22	4,00	4,69	4,78	4,23	4,42	3,87	4,01	4,01	4,01	4,17
Cada DNIS (con my_dnis)	2,0%	6,50	6,42	6,56	6,70	6,30	6,31	6,72	6,37	6,63	6,39	6,62
Cada DNIS (sin my_dnis)		6,37	6,35	6,25	6,30	6,42	6,38	6,35	6,32	6,50	6,31	6,55

Tabla 10. Efecto en búsquedas de campos no indexados de indexar un campo

En conclusión, la introducción de un nuevo campo indexado puede suponer una reducción de hasta el 24% en los tiempos de búsqueda sobre ese campo. Sin embargo, en general, esta mejora rondará tan solo el 10% si la búsqueda filtra únicamente por un valor concreto de dicho campo, cosa que ocurre con frecuencia. El precio a pagar es un aumento del 13% en el espacio de almacenamiento (en general asumible, pues es el recurso más barato en un sistema) y una penalización del 2% en búsquedas de campos no indexados. Este último es el aspecto más importante al evaluar la conveniencia de indexar un campo y con estos datos concretos podría concluirse que si al menos el 17%[22] de las búsquedas que se van a realizar incluyen dicho campo, interesará indexarlo (realmente debería ser algo más frecuente para compensar también el coste en espacio de almacenamiento).

[21] Para tomar tiempos de búsqueda interesa disponer de un conjunto de datos de mayor volumen, para que dichos tiempos aumenten y el resultado se vea menos afectados por otros procesos del sistema que puedan consumir recursos momentáneamente, introduciendo pequeñas desviaciones en el resultado.

[22] Suponiendo, en promedio, búsquedas de la misma duración, la ganancia en tiempo del 10% conseguida mediante la indexación del campo se anula con 5 búsquedas penalizadas un 2% en su duración si no incluyen dicho campo, de lo que se obtiene el umbral 1/6 o 17%.

Conclusiones y trabajo futuro

Splunk es un producto comercial que ofrece una solución sencilla para el manejo de Grandes Datos, sin necesidad de unos conocimientos en profundidad de este campo para diseñar y configurar el modo en que se almacene, diversifique y recupere la información. Así mismo, su facilidad a la hora de escalarlo permite partir de una única instancia que aglutine todas las funciones si el volumen de información a procesar y recuperar es reducido, y que puede ampliarse con nuevas instancias específicas conforme los requerimientos van en aumento.

Tras las pruebas realizadas, puede concluirse que este producto, no sólo no requiere de unos conocimientos avanzados para su despliegue, sino que la inteligencia integrada en el mismo permite hacer uso del mismo sin preocuparse, en un primer momento, por el modo en que se emplee, pues automáticamente optimizará las consultas realizadas sobre el conjunto de datos disponible.

Las funciones ofrecidas por Splunk cumplen con los requisitos fijados por la red de comunicaciones objetivo de este estudio, desde las más básicas como la gestión de redundancia y consistencia de la información, hasta las más específicas en lo referente a las opciones disponibles para la recuperación de la información por parte de los usuarios del sistema, tanto de forma reactiva (bajo demanda) como proactiva (alarmas).

Si bien para el volumen de información indexado y recuperado hubiera bastado con el empleo de una instancia única, se ha preferido el uso de máquinas virtuales simulando una pequeña red de máquinas con instancias especializadas por los siguientes motivos:

- La implementación de Splunk para la red de comunicaciones objeto de este estudio requeriría de múltiples máquinas para gestionar toda la información.
- Familiarizarse con la configuración de una arquitectura clúster de Splunk.
- Comprobar el comportamiento del clúster ante pérdida o sustitución de alguno de los nodos.

Sin embargo el hardware empleado, no preparado para la óptima gestión de máquinas virtuales y sus recursos, ha impedido obtener unos resultados reales en cuanto al efecto de introducir nuevos nodos en la arquitectura: Al no reservar núcleos (Únicamente RAM y almacenamiento) para cada máquina virtual, el procesamiento requerido por cada una se asignaba en cada momento arbitrariamente por el sistema, de modo que siempre se contaba con la misma capacidad de procesamiento (la del hardware empleado), independientemente del número de máquinas virtuales instaladas.

Gracias a los buenos logs de cada tarea generados por Splunk, se pudo comprobar la distribución del trabajo entre cada nodo involucrado, si bien los tiempos de procesamiento absolutos empleados por ellos no resultan realistas por lo indicado arriba.

En caso de querer profundizar en el estudio del rendimiento real de cada máquina, será necesario disponer de un hardware preparado para el despliegue de máquinas virtuales y su dimensionamiento y gestión individualizados, o bien la implementación en una red auténtica con máquinas individuales, tal y como se haría en la implementación final del sistema en la red de comunicaciones de este estudio.

Otra limitación encontrada en la realización de este estudio ha sido el límite de indexación diario de 500MB en la versión gratuita de Splunk. Por este motivo no se llegaron a hacer pruebas con mayores volúmenes de información (como en el caso de los tiempos de búsqueda de campos no indexados con la inclusión o no de un nuevo campo indexado).

Finalmente, podría resultar de interés, si se dispusiese de una plataforma real sobre la que desplegar Splunk, evaluar el rendimiento práctico del producto a la hora de indexar y recuperar información para así poder realizar el dimensionamiento adecuado.
Splunk proporciona un rendimiento teórico en base a unas especificaciones estándar de hardware y número de usuarios[xii], pero dicho estudio permitiría evaluar el grado de estrés al que estarían sometidos los diferentes equipos y estimar en qué punto sería necesario aumentar el número de elementos del clúster.

Planificación y presupuesto

A la hora de llevar a cabo un proyecto de cierta envergadura, resulta importante elaborar una planificación de los procesos que lo compondrán, sus dependencias y las duraciones estimadas de cada uno. De esta manera, conforme se avance en el mismo, podrán compararse los progresos con el calendario previsto, identificando riesgos por posibles retrasos y evitando sobrepasar fechas que pudieran suponer un retraso en la fecha de finalización.

También es necesario elaborar un presupuesto para poder distribuir adecuadamente los recursos económicos disponibles y no correr el riesgo de no disponer para una partida concreta por haber asignado en exceso a otra.

A continuación se presentan las estimaciones de tiempo y costes para este proyecto.

Planificación

El diagrama de Gantt de la Figura 28 muestra las fases en que se ha dividido el proyecto para su elaboración. En esencia, las tareas principales se corresponden con los apartados en que está dividida esta memoria. Si bien ciertas sub-tareas necesitaban de la finalización de otras para dar comienzo, otras hubieran podido realizarse en paralelo, en caso de que hubieran participado más personas en este trabajo.

Nótese que en la fase de *Implementación* existe una subtarea denominada *'Instalación VM y Splunk'*, y otra *'Reinstalación VM y Splunk (Renovación lic)'*. Esta segunda subtarea (En mitad de la fase de *Evaluación*) fue necesaria porque la versión de evaluación gratuita proporcionada por Splunk tiene una validez de 3 meses y las licencias instaladas en las máquinas virtuales estaba próximas a expirar, por lo que fue necesario reinstalar nuevas instancias de Splunk en nuevas máquinas virtuales.

Presupuesto

El coste del proyecto se divide entre el dedicado a recursos humanos y aquel destinado a recursos materiales.

Recursos humanos

Para el cálculo del coste asociado a los recursos humanos podemos considerar una dedicación completa de un Ingeniero durante 4 meses, 40 horas semanales. Se realiza esta consideración puesto que aunque la elaboración del proyecto ha sido, como muestra el diagrama de Gantt, de 9 meses, al tener que compaginarlo con mi actividad laboral, el esfuerzo efectivo habría sido el equivalente a esa duración.

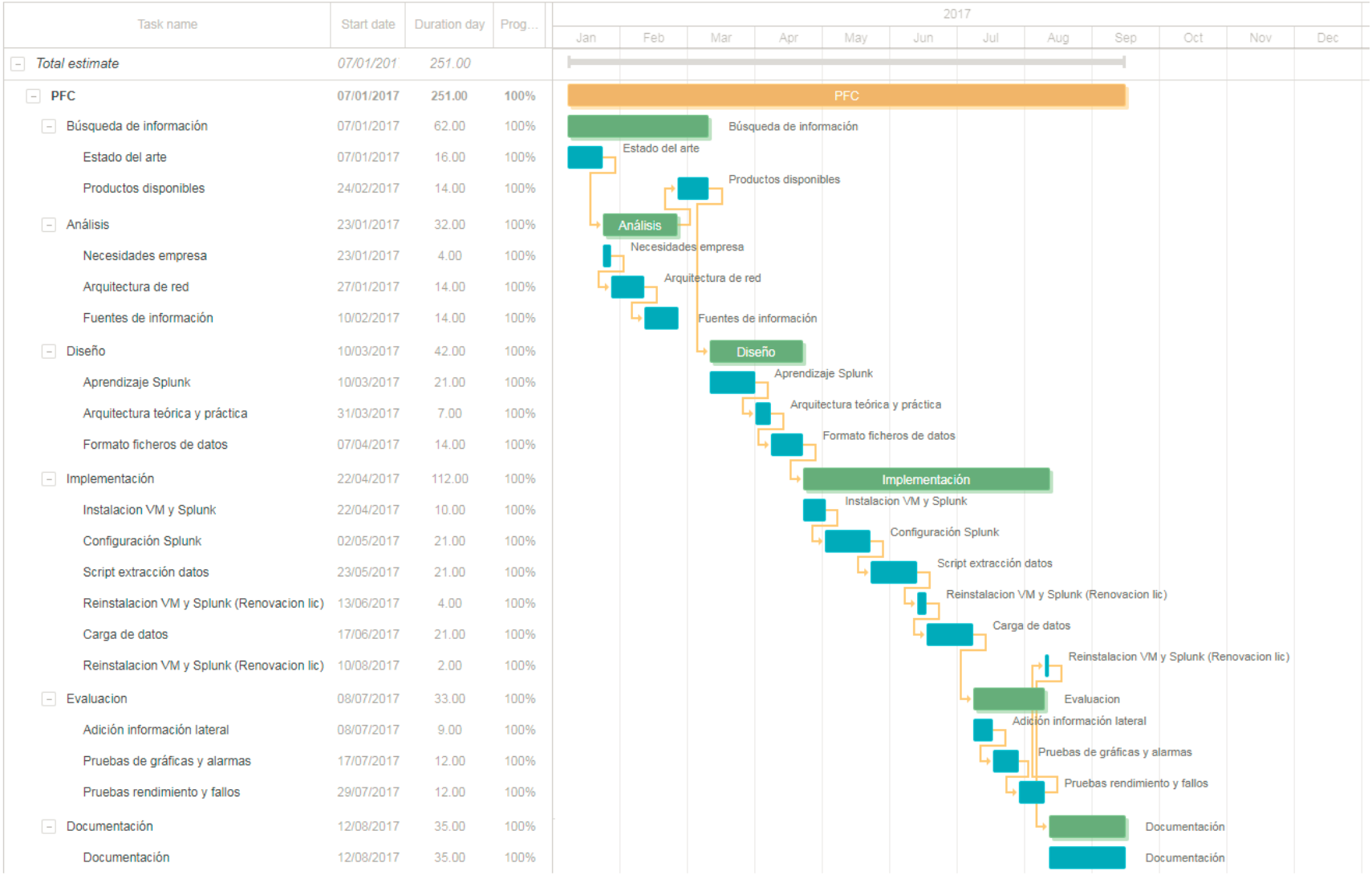


Figura 28. Diagrama de Gantt

La siguiente tabla muestra el desglose de los conceptos empleados para el cálculo de los costes de los recursos humanos, teniendo en cuenta las bases y tipos de cotización de 2017[xiii]:

Coste (€/h)	Dedicación (h/semana)	Coste mensual (€)	Base cotización mensual (€)	Tipo de cotización (%)	Desempleo (%)	Fogasa (%)	F.P. (%)	Coste cuota mensual (€)	Coste total mensual (€)	Duración proyecto (meses)	Coste duración proyecto (€)
15	40	2.400	2.800	23,6	6,7	0,2	0,6	746	3.671	4	14.683

Tabla 11. Coste de los recursos humanos

Para estos cálculos se ha considerado el Grupo de Cotización 1 (Ingenieros y Licenciados), con un contrato de duración determinada a tiempo completo. En la base de cotización mensual se ha tenido en cuenta el prorrateo de dos pagas a lo largo de 12 meses, y el resultado final es un coste de 14.683€.

Recursos materiales

Para la realización de este proyecto se utilizó un ordenador procesador Intel Core I7-6700HQ @ 2.60GHz y 64 GB de RAM, con sistema operativo Windows10, cuyo coste fue de 2.099,62€. Así mismo se adquirió Office 2010 por un precio de 44,62€. Tomando un período de amortización de 5 años para este equipo y considerando un uso de 4 meses, el coste asociado es de:

$$Coste\ Hardware + Software = \frac{2.099{,}62 + 44{,}62}{5 * 12} * 4 = 142{,}95€$$

Por otro lado pueden considerarse otros gastos materiales adicionales como:

- Alquiler de una línea de Fibra Óptica de 50MB, con un coste mensual de 45,45€ (181,80€ en los cuatro meses del proyecto).

- Consumo eléctrico: En la Tabla 12 se desglosa la potencia eléctrica y horas de uso al mes de los aparatos eléctricos utilizados durante el proyecto. Se ha obviado la cuota correspondiente a la contratación de luz, con lo que atendiendo únicamente al coste por energía consumida (0,11 €/ kW h y 5,11% de impuesto), y que el consumo a lo largo de 4 meses fue de 155,68 kW h, resulta un total de 18,48€.

	Ordenador + Router		Iluminación		Aire Acondicionado	
	Potencia (W)		Potencia (W)		Potencia (W)	
	175+12		28		380	
	Horas/dia	kW h mensual	Horas/dia	kW h mensual	Horas/dia	kW h mensual
Marzo	8	29,92	3	1,68	0	0
Abril	8	29,92	3	1,68	0	0
Mayo	8	29,92	2	1,12	1	7,6
Junio	8	29,92	2	1,12	3	22,8

Tabla 12. Consumo eléctrico

Por lo tanto el coste total del proyecto se resume en:

Recurso	Coste
Personal	14.6873,00 €
Ordenador	142,95 €
Conexión a Internet	181,80 €
Consumo eléctrico	18,48 €
Total (sin iva)	15.026,23 €
IVA (21%)	3.155,51 €
TOTAL (iva incluido)	**18.181,74 €**

Tabla 13. Resumen de costes

En conclusión, el coste total ha sido 18.181,74€.

Anexo 1. Script clave-valor

```
#!/usr/bin/perl

# acct_conv_stdout.pl tnc|pTNC|inc|xlc|isdn|vbnac|ntlv|etlv|tlv json|kv [normal|debug|realtime|retime]
# Conversion de registros de accounting a formato JSON o clave-valor
# Parametros de entrada:
#  FILETYPE: (OBLIGATORIO) Cualquiera de los ficheros de accounting conocidos, identificados por su prefijo:
#               tnc, pTNC, inc, xlc, isdn, vbnac, ntlv, etlv, tlv
#  RECTYPE: (OBLIGATORIO) Tipo de fichero de salida
#               json, kv
#  RUMODE: [OPCIONAL] Tipo de ejecución del script
#               normal     Convierte los registros segun los recibe y manda stderr a fichero de log
#               debug      Convierte los registros segun los recibe y manda stderr a pantalla
#               retime     Como <normal>, pero cambia la fecha del registro por la actual
#               realtime   Como <retime>, pero espera a procesar el registro hasta que su hora es menor a la actual

# En sistema alimentado con los datos en tiempo real, la sintaxis completa seria:
#  XLC.: XIMF -all -xlc /u/trx/xlc<yymmdd> | /sourcedata/conv.pl xlc <RECTYPE>
#  TNC.: XIMF -all /u/trx/tnc<yymmdd> | /sourcedata/conv.pl tnc <RECTYPE>
#  ISDN: isdnhaul -e /u/trx/isdn<yymmdd> | /sourcedata/conv.pl isdn <RECTYPE>
#  NTLV: vlahaul -? -ntlv /u/trx/ntlv<yymmdd> | /sourcedata/conv.pl <ntlv|vbnac|etlv> <RECTYPE>
#  TLV.: vlahaul -? -tlv /u/trx/tlv<yymmdd> | /sourcedata/conv.pl tlv <RECTYPE>
#
# Para alimentar offline, se parte de ficheros donde se ha volcado la salida de las herramientas XIMF, isdnhaul o vlahaul
#  Si ademas se escoge RUNMODE retime, los registros del fichero deberian tener una unica fecha
#  Si ademas se escoge RUNMODE realtime, a la restriccion de <retime> hay que añadir preordenar el fichero por hora

# ENTORNO
use strict;       # Fuerza declaracion estricta de variables
use warnings;     # Muestra Warnings
use Switch;       # Necesario para los case
use Time::Local;  # Necesario para localtime

# INICIALIZACION DE VARIABLES
# Parametros de entrada
my $FILETYPE; # Tipo de fichero de entrada
my $RECTYPE;  # Tipo de fichero de salida
my $RUNMODE;  # [Opcional] Tipo de ejecucion -  normal, debug, realtime, retime

# Delimitadores de ficheros de salida
my $RS;     # Comienzo de Registro
my $RE;     # Fin de Registro
my $KD;     # Delimitador de Clave
my $KS;     # Separador Clave - Valor
my $SD;     # Delimitador String
my $FS;     # Delimitador de Campo

my %NAMEHASH;     # hash to hold field name to process mapping
my @FIELD_ARRAY;  # Holds input field array from fixed accounting records
my $TMPSTRING;    # Temporary String Variable used for format strings
my $PSTRING;      # Print String Variable used for format strings
my $PSTRINGDID;
my $PSTRINGPUERTO;
my $PSTRINGPCON;
my $SALIDA='';

my $LINEA;      # Linea leida
my $PRIM3;      # 3 primeros caracteres de la linea leida
```

```
my $NOMBRE='';  # Nombre del campo de VLAHAUL
my $VALOR='';   # Valor del campo de VLAHAUL
my $TMP;        # Variable temporal
my $TMPVAL;     # Variable temporal valor
my $OSRCA;      # OSRCA
my $ODP;        # ODP

# Re-Timing
my $acct_time;         # Epoch del registro
my $acct_epochday;     # Epoch del inicio del dia del registro (00:00:00)
my $local_time;        # Epoch del registro actualizado a hoy: Fecha de hoy + Hora del Registro
my $local_epochday;    # Epoch del inicio del dia de hoy (00:00:00)
my @local_time_arr;    # Array del registro actualizado a hoy
my $retime;            # Fecha que sobreescribir para el dia de hoy: yymmdd. Un registro con yymmdd-1 sera ayer, etc
my $retime_epochday;   # Epoch del inicio del dia de referencia para retime
my $retime_offset;     # Diferencia entre los inicios de dia de la fecha de referencia y la del registro de accounting actual

# Array para devolver los valores numericos de los meses a partir de los valores devueltos por las funciones de fecha
#  y otras variables relacionadas.
my @months = ("01","02","03","04","05","06","07","08","09","10","11","12");
my $sec;
my $min;
my $hour;
my $day;
my $month;
my $year;



# Definición del Hash para procesar los registros de la familia TLV
# Si un campo no existe, o está comentado, dicho campo se ignorará

# Tipo 1: Para DUR, que está en décimas de segundo
$NAMEHASH{'DUR'} = 1;

# Tipo 2: Para SSTS, un epoch, y que debe ser el primer el primer campo en la salida
$NAMEHASH{'SSTS'} = 2;

# Tipo 3: Numérico, obtenido de un Hex
$NAMEHASH{'RFMT'} = 3;
$NAMEHASH{'TOS'} = 3;
$NAMEHASH{'TXB'} = 3;
$NAMEHASH{'RXB'} = 3;
$NAMEHASH{'MTX'} = 3;
$NAMEHASH{'MRX'} = 3;
$NAMEHASH{'TXB'} = 3;
$NAMEHASH{'TXP'} = 3;
$NAMEHASH{'RXP'} = 3;
$NAMEHASH{'RXP'} = 3;
$NAMEHASH{'MTT'} = 3;
$NAMEHASH{'SPD'} = 3;
$NAMEHASH{'OTN'} = 3;
$NAMEHASH{'ODN'} = 3;
$NAMEHASH{'BAUD'} = 3;
$NAMEHASH{'COMP'} = 3;
$NAMEHASH{'ECP'} = 3;
$NAMEHASH{'SRCPRT'} = 3;
$NAMEHASH{'DSTPRT'} = 3;
$NAMEHASH{'ODP'} = 3;
$NAMEHASH{'UDIP'} = 3;
$NAMEHASH{'OAT'} = 3;
$NAMEHASH{'GWI'} = 3;
```

```
$NAMEHASH{'OND'} = 3;
$NAMEHASH{'NID'} = 3;
$NAMEHASH{'SNID'} = 3;
$NAMEHASH{'DID'} = 3;
$NAMEHASH{'HID'} = 3;
$NAMEHASH{'CSN'} = 3;
$NAMEHASH{'OCN'} = 3;
$NAMEHASH{'HOST_CALL_ATTEMPTS'} = 3;
$NAMEHASH{'CNTRY'} = 3;
$NAMEHASH{'PSKF'} = 3;
$NAMEHASH{'RRP_TRAN'} = 3;
$NAMEHASH{'NDID'} = 3;
$NAMEHASH{'CLICK'} = 3;
$NAMEHASH{'OVR_SCN'} = 3;
$NAMEHASH{'DARC'} = 3;
$NAMEHASH{'MSDUR'} = 3;
$NAMEHASH{'CST'} = 3;
$NAMEHASH{'CEV'} = 3;
$NAMEHASH{'TRT'} = 3;
$NAMEHASH{'AXT'} = 3;
$NAMEHASH{'UTXB'} = 3;
$NAMEHASH{'URXB'} = 3;
$NAMEHASH{'SEF'} = 3;
$NAMEHASH{'SSLCIPHER'} = 3;
$NAMEHASH{'SSL_HNDSHK_TIME'} = 3;
$NAMEHASH{'PIN_XLATE'} = 3;
$NAMEHASH{'CONV_FROM_FIX'} = 3;
$NAMEHASH{'NII'} = 3;
$NAMEHASH{'SESSID'} = 3;
$NAMEHASH{'X509_ENROLL_RESP'} = 3;
$NAMEHASH{'RTR'} = 3;
$NAMEHASH{'PHYSID'} = 3;
$NAMEHASH{'TNIP_USESS_ID'} = 3;
$NAMEHASH{'PAN_CLEAR'} = 3;
$NAMEHASH{'PAN_ENCRYPTED'} = 3;
$NAMEHASH{'PAN_TOKENIZED'} = 3;
$NAMEHASH{'RED_REQS'} = 3;
$NAMEHASH{'MSDUR_FILER'} = 3;
$NAMEHASH{'MSDUR_FILEP'} = 3;
$NAMEHASH{'MSDUR_FILEU'} = 3;
$NAMEHASH{'MSDUR_RED'} = 3;
$NAMEHASH{'FID'} = 3;
$NAMEHASH{'DTOK'} = 3;
$NAMEHASH{'FCON'} = 3;

# Tipo 4: Un String con indicador de longitud delante
$NAMEHASH{'DNIS'} = 4;
$NAMEHASH{'ANI'} = 4;
$NAMEHASH{'TID'} = 4;
$NAMEHASH{'MID'} = 4;
$NAMEHASH{'TAID'} = 4;
$NAMEHASH{'IPCSI'} = 4;
$NAMEHASH{'BNSWR'} = 4;
$NAMEHASH{'TMID'} = 4;
$NAMEHASH{'MID0'} = 4;
$NAMEHASH{'TNIP_ID'} = 4;
$NAMEHASH{'HOST_ADDR'} = 4;
$NAMEHASH{'HOST_RSP_TIME'} = 4;
$NAMEHASH{'HOST_NUA'} = 4;
$NAMEHASH{'BID'} = 4;
$NAMEHASH{'TPE_NUMBER'} = 4;
$NAMEHASH{'TASA'} = 4;
```

```
$NAMEHASH{'TPE_SERIAL_NUMBER'} = 4;
$NAMEHASH{'TXN_TYPE'} = 4;
$NAMEHASH{'POS_DEVICE_ID'} = 4;
$NAMEHASH{'SWR'} = 4;
$NAMEHASH{'POS_IDSA'} = 4;
$NAMEHASH{'CALL_SETUP_TIME'} = 4;
$NAMEHASH{'CALL_REF'} = 4;
$NAMEHASH{'SOFTWARE'} = 4;
$NAMEHASH{'VERSION'} = 4;
$NAMEHASH{'TASKNAME'} = 4;
$NAMEHASH{'DSAD'} = 4;
$NAMEHASH{'PROCID'} = 4;
$NAMEHASH{'DNIS_SUBADDR'} = 4;
$NAMEHASH{'PCON_SC'} = 4;
$NAMEHASH{'PCON_PCSC'} = 4;
$NAMEHASH{'DATA'} = 4;
$NAMEHASH{'HOST_CALL_DURATION'} = 4;
$NAMEHASH{'SIM_ID'} = 4;
$NAMEHASH{'CMR'} = 4;
$NAMEHASH{'DATA'} = 4;
$NAMEHASH{'FDI'} = 4;
$NAMEHASH{'HOST_INFO_EVF'} = 4;
$NAMEHASH{'MEF'} = 4;
$NAMEHASH{'POS_MANUFACTURER'} = 4;
$NAMEHASH{'REDIR'} = 4;
$NAMEHASH{'RSRV2'} = 4;
$NAMEHASH{'RTCAR'} = 4;
$NAMEHASH{'X509_ENROLL_CODE'} = 4;
$NAMEHASH{'X509_EXPIRE'} = 4;
$NAMEHASH{'X509_ISSUER_DN'} = 4;
$NAMEHASH{'X509_SERNUM'} = 4;
$NAMEHASH{'IEAT'} = 4;
$NAMEHASH{'CID'} = 4;
$NAMEHASH{'ZKEG'} = 4;
$NAMEHASH{'VCAR'} = 4;
$NAMEHASH{'ACAR'} = 4;
$NAMEHASH{'BCAR'} = 4;
$NAMEHASH{'ZCAR'} = 4;
$NAMEHASH{'PCAR'} = 4;
$NAMEHASH{'XCAR'} = 4;
$NAMEHASH{'ICAR'} = 4;
$NAMEHASH{'ISTW'} = 4;
$NAMEHASH{'RCAR'} = 4;
$NAMEHASH{'BIN'} = 4;
$NAMEHASH{'HOST_DISC_CODE'} = 4;
$NAMEHASH{'FILENAME'} = 4;
$NAMEHASH{'USERNAME'} = 4;
$NAMEHASH{'TSN'} = 4;
$NAMEHASH{'TSPID'} = 4;
$NAMEHASH{'CORRELATION_ID'} = 4;
$NAMEHASH{'TXN_NO'} = 4;

# Tipo 5: ACNAs, para los que debe verificarse si su valor es 'Unknown' o '#'
$NAMEHASH{'ACNA'} = 5;

# Tipo 6: Códigos de terminación (Strings de longitud fija)
$NAMEHASH{'CEF'} = 6;
$NAMEHASH{'FGS'} = 6;
$NAMEHASH{'CSDIAG'} = 6;
$NAMEHASH{'FEF'} = 6;
$NAMEHASH{'SSLVERSION'} = 6;
```

```
# Tipo 7: Direcciones IP
$NAMEHASH{'UDESTA'} = 7;
$NAMEHASH{'SRCIP'} = 7;
$NAMEHASH{'DSTIP'} = 7;
$NAMEHASH{'OSRCA'} = 7;
$NAMEHASH{'ODESTA'} = 7;

# Tipo 8: Resto de campos de fecha/hora
$NAMEHASH{'SETS'} = 8;
$NAMEHASH{'OHTS'} = 8;
$NAMEHASH{'TDATS'} = 8;
$NAMEHASH{'HDATS'} = 8;

# Tipo 9: Campos con codificación BCD, como direcciones IP con indicador de longitud
$NAMEHASH{'DADRP'} = 10;      # DADRP is SET 10 (hex) at this time since some code ouputs it as a string
$NAMEHASH{'SADRP'} = 9;

# Tipo 10: Campox Hex que deben mantenerse como Hex, con indicador de longitud
$NAMEHASH{'SSL_CIPHER_LIST'} = 10;
$NAMEHASH{'MSG_TYPE'} = 10;
$NAMEHASH{'MACID'} = 10;

# Tipo 11: ASCII sin indicador de longitud
$NAMEHASH{'MSG_SPEC_VER'} = 11;     # Aparentemente un dígito entero ASCII
$NAMEHASH{'NRMLTOKNPROC_STATUS'} = 11;  # Aparentemente Y o N

# Tipo 100: Ignorar
$NAMEHASH{'TAGSET'} = 100;
$NAMEHASH{'TYPE'} = 100;

# FIN DE INICIALIZACION DE VARIABLES



# Parametros de entrada
$FILETYPE = $ARGV[0];
$RECTYPE = $ARGV[1];
$RUNMODE = $ARGV[2];
if ( @ARGV > 3 && ($RUNMODE eq 'retime' || $RUNMODE eq 'realtime') ) { $retime = $ARGV[3] }
else { $retime = "" }

# Fijar $RUNMODE a 'normal' si no se introdujo
if (not defined $RUNMODE) {
  $RUNMODE='normal'
}

# Comprobar que se introdujeron los parámetros obligatorios
die "Usage: | ./acct_conv_stdout.pl tnc|pTNC|xlc|inc|isdn|tlv|etlv|ntlv|vbnac|ntlv json|kv [normal|debug|realtime|retime]" if ((not length $FILETYPE) || (not length $RECTYPE));



# Activar auto-flush en STDERR y STDOUT para evitar buffering en la salida
select(STDERR);
$| = 1;
select(STDOUT);
$| = 1;

# Si RUNMODE no es debug
if ($RUNMODE ne 'debug') {
  # Enviar los errores del script a un fichero de acct_conv_stdout.filetype.YYMMDD.log
  ($sec, $min, $hour, $day, $month, $year) = (gmtime())[0,1,2,3,4,5];
  open(LOG, ">>/u/gams/scripts/acct_conv_stdout" . "." . $FILETYPE . "." . substr(($year+1900),-2) . $months[$month] . '0' x (2 - length $day) . $day . ".log");
```

```
  *STDERR = *LOG;
}

# Delimitadores de campo/string para RECTYPE json
if ($RECTYPE eq 'json') {
  $RS = '{';      #Record Start
  $RE = '}';      #Record End
  $KD = '"';      #Key Delimeter
  $KS = ':';      #Key Value Separator
  $SD = '"';      #String Delimeter
  $FS = ',';      #Field Separator
}

# Delimitadores de campo/string para RECTYPE kv
elsif ($RECTYPE eq 'kv')  {
  $RS = '';     #Record Start
  $RE = '';     #Record End
  $KD = '';     #Key Delimeter
  $KS = '=';      #Key Value Separator
  $SD = '"';      #String Delimeter
  $FS = ' ';      #Field Separator
}
# Fallar en caso contrario
else {
  die "Bad RECTYPE.\nUsage: | ./acct_conv_stdout.pl tnc|pTNC|xlc|inc|isdn|tlv|etlv|ntlv|vbnac|ntlv json|kv [normal|debug|realtime|retime]"
}


# Procesamiento de registros de accounting
# Fichero isdn
if ($FILETYPE eq 'isdn') {

  # Definir el formato de impresión del String de salida
  $TMPSTRING = $RS . $KD . 'SSTS' . $KD . $KS . $SD . '%s:00' . $SD . $FS . $KD . 'DUR' . $KD . $KS . '%d' . $FS . $KD . 'RXB' . $KD . $KS . '%d' . $FS . $KD . 'TXB' . $KD . $KS . '%d' . $FS .
$KD . 'OND' . $KD . $KS . '%d' . $FS . $KD . 'RXP' . $KD . $KS . '%d' . $FS . $KD . 'TXP' . $KD . $KS . '%d' . $FS . $KD . 'HID' . $KD . $KS . '%d' . $FS . $KD . 'SEF' . $KD . $KS . '%d' . $FS .
$KD . 'GWI' . $KD . $KS . '%d' . $FS . $KD . 'CSN' . $KD . $KS . '%d' . $FS . $KD . 'CSDIAG' . $KD . $KS . $SD . '%s' . $SD;
  # Bucle leyendo linea por linea: isdnhaul (-e) separa los campos con comas
  while (<STDIN>) {

    # Quitar EOL
    chomp;
    # Quitar comillas
    $_ =~ s/"//g;
    # Transformar en array
    @FIELD_ARRAY = split ",", $_;

    # Omitir registro si esta corrupto
    if ("01/01/70" ne $FIELD_ARRAY[0]){

      # Initial setup of the Print String for printf
      $PSTRING = $TMPSTRING;

      # La mayoria de registros isdn son IP, pero también hay X.25, así que hay que determinar el tipo para
      # las direcciones origen y destino.
      # Para ello, se rellenará con 0s a la izquierda hasta completar 16 dígitos.
      # Si los 12 primeros digitos (4 grupos de 3 digitos) constituyen una IP válida, y los 4 siguientes son Hex,
      # se considerará que se trata de una Dirección IP (que se formateará como ###.###.###.###) mas Puerto.
      # En caso contrario, el campo completo se mostrará tal cual como dirección (origen/destino segun corresponda).

      # Dirección origen
      # Pasar a variable temporal
      $TMP = $FIELD_ARRAY[12];
```

```
# Quitar espacios, si los tuviese
$TMP =~ s/\s*//g;

# Si la longitud de la dirección origen es > 0
if (length($TMP) > 0) {

  # Reformatear a 16 caracteres de longitud, rellenando con 0s a la izquierda
  $TMP = sprintf("%016s", $TMP);

  # Si es Dirección IP + Puerto
  if ($TMP =~ m/^(\d{3})(\d{3})(\d{3})(\d{3})([0-9ABCDEF]{4})$/ ) {
    if ($1 <= 255 && $2 <= 255 && $3 <= 255 && $4 <= 255) {
      # Extraer campos Dirección IP y Puerto y añadirlos a string de salida
      $OSRCA = sprintf("%d.%d.%d.%d",$1,$2,$3,$4);
      $ODP = hex $5;

      $PSTRING = $PSTRING . $FS . $KD . 'OSRCA' . $KD . $KS . $SD . $OSRCA . $SD . $FS . $KD . 'ODP' . $KD . $KS . $ODP;
    }
    # En caso contrario sacar Dirección Origen tal cual en string de salida
    else {
      $PSTRING = $PSTRING . $FS . $KD . 'OSRCA' . $KD . $KS . $SD . $TMP . $SD;
    }
  }
  # En caso contrario sacar Dirección Origen tal cual en string de salida
  else {
    $PSTRING = $PSTRING . $FS . $KD . 'OSRCA' . $KD . $KS . $SD . $TMP . $SD;
  }
}

# Dirección destino
# Pasar a variable temporal
$TMP = $FIELD_ARRAY[13];

# Quitar espacios, si los tuviese
$TMP =~ s/\s*//g;

# Si la longitud de la dirección destino es > 0
if (length($TMP) > 0) {

  # Reformatear a 16 caracteres de longitud, rellenando con 0s a la izquierda
  $TMP = sprintf("%016s", $TMP);

  # Si es Dirección IP + Puerto
  if ($TMP =~ m/^(\d{3})(\d{3})(\d{3})(\d{3})([0-9ABCDEF]{4})$/ ) {
    if ($1 <= 255 && $2 <= 255 && $3 <= 255 && $4 <= 255) {
      # Extraer campos Dirección IP y Puerto y añadirlos a string de salida
      $OSRCA = sprintf("%d.%d.%d.%d",$1,$2,$3,$4);
      $ODP = hex $5;

      $PSTRING = $PSTRING . $FS . $KD . 'UDESTA' . $KD . $KS . $SD . $OSRCA . $SD . $FS . $KD . 'UDIP' . $KD . $KS . $ODP;
    }
    # En caso contrario sacar Dirección Destino tal cual en string de salida
    else {
      $PSTRING = $PSTRING . $FS . $KD . 'UDESTA' . $KD . $KS . $SD . $TMP . $SD;
    }
  }
  # En caso contrario sacar Dirección Destino tal cual en string de salida
  else {
    $PSTRING = $PSTRING . $FS . $KD . 'UDESTA' . $KD . $KS . $SD . $TMP . $SD;
  }
}
```

```
      # Completar string de salida
      $PSTRING = $PSTRING . $RE;

      # Salida
      # Se añade :00 en el campo SSTS, pues en datos isdn no se especifican los segundos.

      printf($PSTRING  .  "\n",  $FIELD_ARRAY[0],  $FIELD_ARRAY[1],  $FIELD_ARRAY[2],  $FIELD_ARRAY[3],  $FIELD_ARRAY[4],  $FIELD_ARRAY[5],  $FIELD_ARRAY[6],  $FIELD_ARRAY[7],  $FIELD_ARRAY[8],
$FIELD_ARRAY[9], $FIELD_ARRAY[10], $FIELD_ARRAY[11]);
    }
  }
}

# Fichero tnc, pTNC (TNC puro) o inc ---
elsif (($FILETYPE eq 'tnc') || ($FILETYPE eq 'pTNC') || ($FILETYPE eq 'inc')) {

  # Definicion de los formatos de salida (cuatro variantes).
       # Se reformatea la fecha para que sea igual a la de los formatos isdn y vlahaul
  # NOTAS
  # Campos comunes:
  # 1) SSTS; 2) RFMT; 4) DUR; 5) OND;  11) DNIS; 12) TXB; 12) RXB; 13) FGS
  # Excepciones:
  # No en PCON
  # 3) TOS; 6) OCN; 7) OTN; 8) ODN; 9) SPD; 10) ANI; 14) NID 15) SNID
  # Solo registros con digito DID en campo DNIS, que se calcula segun contenido
  # 16) DID
  # Solo registros con un ODP en DNIS que se calcula en base al contenido
  # 16) ODP

  # Registros estandar
  $PSTRING = $RS . $KD . 'SSTS' . $KD . $KS . $SD . '%s/%s/%s %s' . $SD . $FS . $KD . 'RFMT' . $KD . $KS . '%d' . $FS . $KD . 'TOS' . $KD . $KS . '%d' . $FS . $KD . 'DUR' . $KD . $KS . '%.1f' .
$FS . $KD . 'OND' . $KD . $KS . '%d' . $FS . $KD . 'OCN' . $KD . $KS . '%d' . $FS . $KD . 'OTN' . $KD . $KS . '%d' . $FS . $KD . 'ODN' . $KD . $KS . '%d' . $FS . $KD . 'SPD' . $KD . $KS . '%d' .
$FS . $KD . 'ANI' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'DNIS' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'TXB' . $KD . $KS . '%d' . $FS . $KD . 'RXB' . $KD . $KS . '%d' . $FS . $KD .
'FGS' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'NID' . $KD . $KS . '%d' . $FS . $KD . 'SNID' . $KD . $KS . '%d' . $RE;

  # Registros con Dial Indicator
  $PSTRINGDID = $RS . $KD . 'SSTS' . $KD . $KS . $SD . '%s/%s/%s %s' . $SD . $FS . $KD . 'RFMT' . $KD . $KS . '%d' . $FS . $KD . 'TOS' . $KD . $KS . '%d' . $FS . $KD . 'DUR' . $KD . $KS . '%.1f'
. $FS . $KD . 'OND' . $KD . $KS . '%d' . $FS . $KD . 'OCN' . $KD . $KS . '%d' . $FS . $KD . 'OTN' . $KD . $KS . '%d' . $FS . $KD . 'ODN' . $KD . $KS . '%d' . $FS . $KD . 'SPD' . $KD . $KS . '%d'
. $FS . $KD . 'ANI' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'DNIS' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'TXB' . $KD . $KS . '%d' . $FS . $KD . 'RXB' . $KD . $KS . '%d' . $FS . $KD .
'FGS' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'NID' . $KD . $KS . '%d' . $FS . $KD . 'SNID' . $KD . $KS . '%d' . $FS . $KD . 'DID' . $KD . $KS . '%s' . $RE;

  # Registros con Puerto
  $PSTRINGPUERTO = $RS . $KD . 'SSTS' . $KD . $KS . $SD . '%s/%s/%s %s' . $SD . $FS . $KD . 'RFMT' . $KD . $KS . '%d' . $FS . $KD . 'TOS' . $KD . $KS . '%d' . $FS . $KD . 'DUR' . $KD . $KS .
'%.1f' . $FS . $KD . 'OND' . $KD . $KS . '%d' . $FS . $KD . 'OCN' . $KD . $KS . '%d' . $FS . $KD . 'OTN' . $KD . $KS . '%d' . $FS . $KD . 'ODN' . $KD . $KS . '%d' . $FS . $KD . 'SPD' . $KD . $KS
. '%d' . $FS . $KD . 'ANI' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'DNIS' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'TXB' . $KD . $KS . '%d' . $FS . $KD . 'RXB' . $KD . $KS . '%d' . $FS
. $KD . 'FGS' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'NID' . $KD . $KS . '%d' . $FS . $KD . 'SNID' . $KD . $KS . '%d' . $FS . $KD . 'ODP' . $KD . $KS . '%d' . $RE;

  # Registros de PCON
  $PSTRINGPCON = $RS . $KD . 'SSTS' . $KD . $KS . $SD . '%s/%s/%s %s' . $SD . $FS . $KD . 'RFMT' . $KD . $KS . '%d' . $FS . $KD . 'DUR' . $KD . $KS . '%.1f' . $FS . $KD . 'OND' . $KD . $KS .
'%d' . $FS . $KD . 'DNIS' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'TXB' . $KD . $KS . '%d' . $FS . $KD . 'RXB' . $KD . $KS . '%d' . $FS . $KD . 'FGS' . $KD . $KS . $SD . '%s' . $SD . $RE;

  # Bucle leyendo lineas completas de STDIN
  while (<STDIN>) {

    # Quitar EOL
    chomp;
    # Quitar comillas
    $_ =~ s/"//g;
    # Transformar en array
    @FIELD_ARRAY = split " ", $_;

    # Omitir registro si esta corrupto
```

```
    if (1970 != substr($FIELD_ARRAY[2],0,4) ) {

      # Reformateo de fecha
      if ($RUNMODE eq 'realtime' || $RUNMODE eq 'retime') {
              # Inicializa fecha de referencia para retime
              if ( ! $retime_epochday ) {
                      # Si no se introdujo una en llamada a script, usa la primera que encuentra en el fichero
                      if ( ! $retime ) {
                              $retime_epochday=timegm(0,0,0,substr($FIELD_ARRAY[2],8,2),substr($FIELD_ARRAY[2],5,2)-1,substr($FIELD_ARRAY[2],0,4)-1900);
                      }
                      # Si se introdujo una fecha en la llamada al script, la usa como referencia para retime
                      else {
            $retime_epochday = timegm( 0,0,0,substr($retime,4),substr($retime,2,2)-1,"1" . substr($retime,0,2) );
                      }
              }
              # Calcula nueva fecha/hora que fijar en registro
              $acct_time=timegm(substr($FIELD_ARRAY[3],6,2),substr($FIELD_ARRAY[3],3,2),substr($FIELD_ARRAY[3],0,2),substr($FIELD_ARRAY[2],8,2),substr($FIELD_ARRAY[2],5,2)-
1,substr($FIELD_ARRAY[2],0,4)-1900);
              $acct_epochday = timegm(0,0,0,substr($FIELD_ARRAY[2],8,2),substr($FIELD_ARRAY[2],5,2)-1,substr($FIELD_ARRAY[2],0,4)-1900);
              $retime_offset = $acct_epochday - $retime_epochday;
              $local_epochday = timegm(0,0,0,(gmtime())[3],(gmtime())[4],(gmtime())[5]);
              $local_time = 0 + $acct_time - $acct_epochday + $local_epochday + $retime_offset;

              # Espera a que hora actual alcance a hora del registro
              while (timegm(gmtime())<$local_time && $RUNMODE eq 'realtime') {
                      # printf "Sleeping: %s - %s - %s %s\n", timegm(gmtime()), $local_time, $FIELD_ARRAY[2], $FIELD_ARRAY[3];
                      sleep(1);
              }

              # Modifica fecha/hora del registro en la salida
              @local_time_arr=gmtime($local_time);
        $FIELD_ARRAY[2]=sprintf "%04d-%02d-%02d",$local_time_arr[5]+1900,$local_time_arr[4]+1,$local_time_arr[3];
        $FIELD_ARRAY[3]=sprintf "%02d:%02d:%02d",$local_time_arr[2],$local_time_arr[1],$local_time_arr[0];
      }

      # Salida
      # Verificar si se trata de tnc, pTNC o inc estandar de 17 campos
      if (@FIELD_ARRAY == 17) {

        # Si el campo ANI tiene 4 o 5 caracteres y es numerico, asumir que es un puerto y añadirlo a su campo
        if ((length($FIELD_ARRAY[10]) < 6) && (length($FIELD_ARRAY[10]) > 3) && ($FIELD_ARRAY[10] =~ /^\d+?$/)) {
          printf($PSTRINGPUERTO . "\n", substr($FIELD_ARRAY[2],5,2), substr($FIELD_ARRAY[2],8,2), substr($FIELD_ARRAY[2],2,2), $FIELD_ARRAY[3], $FIELD_ARRAY[0], $FIELD_ARRAY[1], $FIELD_ARRAY[4],
$FIELD_ARRAY[5], $FIELD_ARRAY[6], $FIELD_ARRAY[7], $FIELD_ARRAY[8], $FIELD_ARRAY[9], $FIELD_ARRAY[10], $FIELD_ARRAY[11], $FIELD_ARRAY[12], $FIELD_ARRAY[13], $FIELD_ARRAY[14], $FIELD_ARRAY[15],
$FIELD_ARRAY[16], $FIELD_ARRAY[10]);
        }

        # Comprobar si el DNIS tiene un digito Dial Indicator, y en ese caso
                                # ponerlo en el campo DID y quitarlo del campo DNIS
        # Regla:
        #   UK.: El DNIS tiene 12 caracteres, y del 2 al 5 son 0800
        #   IRL: El DNIS tiene 11 caracteres, y del 2 al 5 son 1800
        elsif (((length($FIELD_ARRAY[11]) == 12) && (substr($FIELD_ARRAY[11],1,4) eq '0800')) || ((length($FIELD_ARRAY[11]) == 11) && (substr($FIELD_ARRAY[11],1,4) eq '1800'))) {
          printf($PSTRINGDID . "\n", substr($FIELD_ARRAY[2],5,2), substr($FIELD_ARRAY[2],8,2), substr($FIELD_ARRAY[2],2,2), $FIELD_ARRAY[3], $FIELD_ARRAY[0], $FIELD_ARRAY[1], $FIELD_ARRAY[4],
$FIELD_ARRAY[5], $FIELD_ARRAY[6], $FIELD_ARRAY[7], $FIELD_ARRAY[8], $FIELD_ARRAY[9], $FIELD_ARRAY[10], substr($FIELD_ARRAY[11],1), $FIELD_ARRAY[12], $FIELD_ARRAY[13], $FIELD_ARRAY[14],
$FIELD_ARRAY[15], $FIELD_ARRAY[16], substr($FIELD_ARRAY[11],0,1));
        }

        # En caso contrario, son campos estandar
        else {
          printf($PSTRING . "\n", substr($FIELD_ARRAY[2],5,2), substr($FIELD_ARRAY[2],8,2), substr($FIELD_ARRAY[2],2,2), $FIELD_ARRAY[3], $FIELD_ARRAY[0], $FIELD_ARRAY[1], $FIELD_ARRAY[4],
$FIELD_ARRAY[5], $FIELD_ARRAY[6], $FIELD_ARRAY[7], $FIELD_ARRAY[8], $FIELD_ARRAY[9], $FIELD_ARRAY[10], $FIELD_ARRAY[11], $FIELD_ARRAY[12], $FIELD_ARRAY[13], $FIELD_ARRAY[14], $FIELD_ARRAY[15],
$FIELD_ARRAY[16]);
```

```
        }
      }

      # En caso contrario se trata de registros PCON obtenidos de TLV, con menos campos
      else {
        printf($PSTRINGPCON . "\n", substr($FIELD_ARRAY[1],5,2), substr($FIELD_ARRAY[1],8,2), substr($FIELD_ARRAY[1],2,2), $FIELD_ARRAY[2], $FIELD_ARRAY[0], $FIELD_ARRAY[3], $FIELD_ARRAY[4],
$FIELD_ARRAY[6], $FIELD_ARRAY[7], $FIELD_ARRAY[8], $FIELD_ARRAY[9]);
      }
    }
  }
}

# Fichero xlc
elsif ($FILETYPE eq 'xlc') {

  # Definir el formato de impresión del String de salida
  $PSTRING = $RS . $KD . 'SETS' . $KD . $KS . $SD . '%s/%s/%s %s' . $SD . $FS . $KD . 'RFMT' . $KD . $KS . '%d' . $FS . $KD . 'SID' . $KD . $KS . '%d' . $FS . $KD . 'DUR' . $KD . $KS . '%.1f' .
$FS . $KD . 'OND' . $KD . $KS . '%d' . $FS . $KD . 'ODI' . $KD . $KS . '%d' . $FS . $KD . 'ANI' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD . 'DNIS' . $KD . $KS . $SD . '%s' . $SD . $FS . $KD .
'ODN' . $KD . $KS . '%d' . $FS . $KD . 'OBC' . $KD . $KS . '%d' . $FS . $KD . 'SCC' . $KD . $KS . $SD . '%s' . $SD . $RE;
  # Bucle leyendo lineas completas de STDIN
  while (<STDIN>) {

    # Quitar EOL
    chomp;
    # Quitar comillas
    $_ =~ s/"//g;
    # Transformar en array
    @FIELD_ARRAY = split " ", $_;

    # Omitir registro si esta corrupto
    if (1970 != substr($FIELD_ARRAY[3],0,4) ){

      # Convierte a decimal el valor hexadecimal del campo 2
      $FIELD_ARRAY[2] = hex substr($FIELD_ARRAY[2],-2);
      # Salida
      printf($PSTRING . "\n", substr($FIELD_ARRAY[3],5,2), substr($FIELD_ARRAY[3],8,2), substr($FIELD_ARRAY[3],2,2), $FIELD_ARRAY[4], $FIELD_ARRAY[0], $FIELD_ARRAY[2], $FIELD_ARRAY[5],
$FIELD_ARRAY[6], $FIELD_ARRAY[7], $FIELD_ARRAY[10], $FIELD_ARRAY[11], $FIELD_ARRAY[12], $FIELD_ARRAY[13], $FIELD_ARRAY[14]);
    }
  }
}

# Ficheros vbnac, ntlv, tlv, etlv
elsif (($FILETYPE eq 'vbnac') || ($FILETYPE eq 'ntlv') || ($FILETYPE eq 'tlv') || ($FILETYPE eq 'etlv')) {

  # Bucle leyendo lineas completas de STDIN
  while (<STDIN>) {

    # Se pone entrada en una variable y se leen los 3 primeros caracteres (Buscando AFI o  fin de registro)
    $LINEA = $_;
    $PRIM3 = substr $_, 0 ,3;

    # Case en base a 3 primeros caracteres
    switch ($PRIM3) {
      # Si es AFI, procesar clave/valor
      case "AFI" {
        if (length $LINEA > 65) {
          # Nombre del campo esta entre posiciones 4 a 23. Eliminar espacios.
          $NOMBRE = substr $LINEA, 4, 23;
          $NOMBRE =~ s/\s*$//;

          # Valor del campo esta de la posicion 66 hasta el final
          $VALOR = substr $LINEA, 66;
```

```
          # Case, segun familia a que pertenece el campo segun hash definido al principio
          switch ($NAMEHASH{$NOMBRE}) {

            # Case 1: Solo para campo DUR, numero en decisegundos
            case 1 {
              # Quitar espacios, covertir hex->dec, dividir por 10, añadir a string de salida sin delimitadores de string
              $VALOR =~ s/\s*//g;
              $VALOR = hex $VALOR;
              $VALOR = $VALOR / 10;
              $SALIDA = $SALIDA . $KD. $NOMBRE . $KD . $KS . $VALOR . $FS;
            }

            # Case 2: Solo para campo SSTS, hora, que debe ser el primer campo del registro
            case 2 {
              # Quitar espacios, convertir hex->dec, obtener gmt, formatear como fecha y hora
              $VALOR =~ s/\s*//g;
              $VALOR = hex $VALOR;
              ($sec, $min, $hour, $day, $month, $year) = (gmtime($VALOR))[0,1,2,3,4,5];

              # Reformateo de fecha
              if ($RUNMODE eq 'realtime' || $RUNMODE eq 'retime') {
                # Inicializa fecha de referencia para retime
                if ( ! $retime_epochday ) {
                  # Si no se introdujo una en llamada a script, usa la primera que encuentra en el fichero
                  if ( ! $retime ) {
                    $retime_epochday = timegm(0,0,0,$day+1,$month,$year);
                        }
                  # Si se introdujo una fecha en la llamada al script, la usa como referencia para retime
                  else {
                    $retime_epochday = timegm( 0,0,0,substr($retime,4),substr($retime,2,2)-1,"1" . substr($retime,0,2) );
                  }
                }

                # Calcula nueva fecha/hora que fijar en registro
                $acct_time=$VALOR;
                $acct_epochday = timegm(0,0,0,$day,$month,1900+$year);
                    $retime_offset = $acct_epochday - $retime_epochday;
                $local_epochday = timegm(0,0,0,(gmtime())[3],(gmtime())[4],(gmtime())[5]);
                $local_time = 0 + $acct_time - $acct_epochday + $local_epochday + $retime_offset;

                # Espera a que hora actual alcance a hora del registro
                while (timegm(gmtime())<$local_time && $RUNMODE eq 'realtime') {
                  # printf "Sleeping: %s - %s \n", timegm(gmtime()), $local_time;
                  sleep(1);
                }

                # Modifica fecha/hora del registro en la salida
                @local_time_arr=gmtime($local_time);
                $day=$local_time_arr[3];
                $month=$local_time_arr[4];
                $year=$local_time_arr[5];
              }

              $SALIDA = $RS . $KD . $NOMBRE . $KD . $KS . $SD . $months[$month] . '/' . '0' x (2 - length $day) . $day . '/' . substr(($year+1900),-2) . ' ' . '0' x (2 - length $hour) . $hour .
':' . '0' x (2 - length $min) . $min . ':' . '0' x (2 - length $sec)  . $sec . $SD . $FS . $SALIDA;
            }

            # Case 3: Mayoria de campos numericos
            case 3 {
              # Quitar espacios, covertir hex->dec, dividir por 10, añadir a string de salida sin delimitadores de string
              $VALOR =~ s/\s*//g;
```

```
  $VALOR = hex $VALOR;
  $SALIDA = $SALIDA . $KD. $NOMBRE . $KD . $KS . $VALOR . $FS;
}

# Case 4: Mayoria de campos string
case 4 {
                                        # Comenzar en posicion 5, saltandose [hh] de longitud
  $VALOR = substr $VALOR, 5;
                                        # Quitar espacios
  $VALOR =~ s/\s*//g;
                                        # Convertir hex->ASCII
  $VALOR =~ s/([a-fA-F0-9][a-fA-F0-9])/chr(hex($1))/eg;
                                        # Quitar caracteres que no sean ASCII y comillas dobles
  $VALOR =~ tr/\040-\176/ /c;
  $VALOR =~ tr/\000-\037/ /;
  $VALOR =~ s/"//g;
                                        # Quitar espacios iniciales y finales
  $VALOR =~ s/^\s*//;
  $VALOR =~ s/\s*$//;
                                        # Si el resultado tiene longitud mayor a 0, añadir a salida con delimitadores de string
  if (length($VALOR) != 0) {
    $SALIDA = $SALIDA . $KD . $NOMBRE . $KD . $KS . $SD . $VALOR . $SD . $FS;
  }
}

# Case 5: Solo para ACNA, filtrando Unknown y #
case 5 {
                                        # Comenzar en posicion 5, saltandose [hh] de longitud
  $VALOR = substr $VALOR, 5;
                                        # Quitar espacios
  $VALOR =~ s/\s*//g;
                                        # Convertir hex->ASCII
  $VALOR =~ s/([a-fA-F0-9][a-fA-F0-9])/chr(hex($1))/eg;
                                        # Quitar caracteres que no sean ASCII y comillas dobles
  $VALOR =~ tr/\040-\176/ /c;
  $VALOR =~ tr/\000-\037/ /;
  $VALOR =~ s/"//g;
                                        # Quitar espacios iniciales y finales
  $VALOR =~ s/^\s*//;
  $VALOR =~ s/\s*$//;
                                        # Si el resultado tiene longitud mayor a 0, y no es "Unknown" ni "#",
  # añadir a salida con delimitadores de string
  if ((length($VALOR) != 0) && ($VALOR ne 'Unknown') && ($VALOR ne '#') && ($VALOR ne '0')) {
    $SALIDA = $SALIDA . $KD . $NOMBRE . $KD . $KS . $SD . $VALOR . $SD . $FS;
  }
}

# Case 6: Strings que no tienen cabecera de longitud y dan un string hexadecimal
case 6 {
                                        # Quitar espacios
  $VALOR =~ s/\s*//g;
  # Añadir a salida con delimitadores de string
  $SALIDA = $SALIDA . $KD . $NOMBRE . $KD . $KS . $SD . $VALOR . $SD . $FS;
}

# Case 7: Campos de direcciones IP
case 7 {
  # Añadir puntos entre los octetos, transformando a decimal
  $VALOR =~ m/^\s+([0-9a-fA-F]{2})\s+([0-9a-fA-F]{2})\s+([0-9a-fA-F]{2})\s+([0-9a-fA-F]{2})\s+$/;
  $TMPVAL = sprintf("%d.%d.%d.%d",hex($1), hex($2), hex($3), hex($4));
  $SALIDA = $SALIDA . $KD . $NOMBRE . $KD . $KS . $SD . $TMPVAL . $SD . $FS;
}
```

```
            # Case 8: Otros campos de fecha/hora
            case 8 {
              # Quitar espacios, convertir hex->dec, obtener gmt, formatear como fecha y hora
              $VALOR =~ s/\s*//g;
              $VALOR = hex $VALOR;
              ($sec, $min, $hour, $day, $month, $year) = (gmtime($VALOR))[0,1,2,3,4,5];
              $SALIDA = $SALIDA . $KD . $NOMBRE . $KD . $KS . $SD . $months[$month] . '/' . '0' x (2 - length $day) . $day . '/' . substr(($year+1900),-2) . ' ' . '0' x (2 - length $hour) .
$hour . ':' . '0' x (2 - length $min) . $min . ':' . '0' x (2 - length $sec)  . $sec . $SD . $FS;
            }

            # Case 9: Campos de texto BCD, con cabecera de longitud, pero de los que solo se extrae el string
            case 9 {
                                                  # Comenzar en posicion 5, saltandose [hh] de longitud
              $VALOR = substr $VALOR, 5;
              # Mostrar cada octeto de los valores hexadecimales como decimal, separados por puntos .
              $VALOR =~ m/^\s+([0-9a-fA-F]{2})\s+([0-9a-fA-F]{2})\s+([0-9a-fA-F]{2})\s+([0-9a-fA-F]{2})\s+$/;
              $TMPVAL = sprintf("%d.%d.%d.%d",hex($1), hex($2), hex($3), hex($4));
              $SALIDA = $SALIDA . $KD . $NOMBRE . $KD . $KS . $SD . $TMPVAL . $SD . $FS;
            }

            # Case 10: Campos hexadecimales que deben mantenerse como hexadecimales, pero con cabecera de longitud
            case 10 {
                                                  # Comenzar en posicion 5, saltandose [hh] de longitud
              $VALOR = substr $VALOR, 5;
                                                  # Quitar espacios
              $VALOR =~ s/\s*//g;
                                                  # Si el resultado tiene longitud mayor a 0, añadir a salida con delimitadores de string
              if (length($VALOR) != 0) {
                $SALIDA = $SALIDA . $KD . $NOMBRE . $KD . $KS . $SD . $VALOR . $SD . $FS;
              }
            }

            # Case 11: ASCII sin cabecera de longitud
            case 11 {
                                                  # Quitar espacios
              $VALOR =~ s/\s*//g;
                                                  # Convertir hex->ASCII
              $VALOR =~ s/([a-fA-F0-9][a-fA-F0-9])/chr(hex($1))/eg;
                                                  # Quitar caracteres que no sean ASCII y comillas dobles
              $VALOR =~ tr/\040-\176/ /c;
              $VALOR =~ tr/\000-\037/ /;
              $VALOR =~ s/"//g;
                                                  # Quitar espacios iniciales y finales
              $VALOR =~ s/^\s*//;
              $VALOR =~ s/\s*$//;
                                                  # Si el resultado tiene longitud mayor a 0, añadir a salida con delimitadores de string
              if (length($VALOR) != 0) {
                $SALIDA = $SALIDA . $KD . $NOMBRE . $KD . $KS . $SD . $VALOR . $SD . $FS;
              }
            }

          }
        }
      }
      # Si es "===", hemos llegado al final del registro
      case "===" {

        # Nos cercioramos de que al menos exista el campo de fecha/hora del registro, para evitar registros corruptos
        if ($SALIDA =~ /^(..)?SSTS.*/) {

          # Sacamos el string de salida por STDOUT
```

```
          print $SALIDA . $RE . "\n";
        }
        # Limpiamos string de salida
        $SALIDA = '';

      ;;

      }
    }
  }
}
```

Anexo 2. Panel con diagrama de burbujas

A continuación se muestra el código correspondiente al panel que mostraba un diagrama de burbujas en función de los bytes recibidos y transmitidos para cada operación en un intervalo determinado (apartado Visualización y Desglose: Drill-down).

```
<form>
  <init>
    <set token="wWhen">x</set>
    <set token="wRXB">-1</set>
    <set token="wTXB">-1</set>
  </init>
  <label>Burbujas según los bytes transmitidos y recibidos</label>
  <description>Tráfico de estudio: Numeración 4-Dígitos Jazztel</description>
  <search id="s1">
    <query>index=my_index NID=34 DNIS="000001*" | eval when="1" | kmeans k=11 RXB,TXB, when
       | append [ search [|gentimes start=-1 | addinfo | eval earliest=info_min_time - 86400 | eval latest=info_max_time - 86400 | table earliest latest | format ("","","","","","") ]
index=my_index NID=34 DNIS="000001*" | eval when="0" | kmeans k=11, RXB, TXB, when ]
      | replace "1" with "Hoy" in when | replace "0" with "Ayer" in when | rename centroid_RXB AS centroide_RXB, centroid_TXB as centroide_TXB</query>
    <earliest>$myDate.earliest$</earliest>
    <latest>$myDate.latest$</latest>
  </search>
  <fieldset submitButton="false">
    <html>Para eje por defecto, escriba "auto"</html>
    <input type="text" token="maxRXB">
      <label>Max RXB (Eje X)</label>
      <default>auto</default>
    </input>
    <input type="text" token="maxTXB">
      <label>Max TXB (Eje Y)</label>
      <default>auto</default>
    </input>
    <input type="time" token="myDate">
      <label>Período</label>
      <default>
        <earliest>1497884400</earliest>
        <latest>1497888000</latest>
      </default>
    </input>
  </fieldset>
  <row>
    <panel>
      <single>
        <search>
          <query>|gentimes start=-1 | addinfo | convert ctime(*) | eval fechahora="Período de referencia desde " . info_min_time . " a " . info_max_time . ". Comparado con mismo período del día
anterior." | table fechahora</query>
          <earliest>$myDate.earliest$</earliest>
          <latest>$myDate.latest$</latest>
        </search>
        <option name="drilldown">none</option>
        <option name="beforeLabel"> </option>
        <option name="linkView">search</option>
        <option name="afterLabel">.</option>
        <option name="colorBy">value</option>
        <option name="colorMode">none</option>
        <option name="numberPrecision">0</option>
```

```
      <option name="showSparkline">1</option>
      <option name="showTrendIndicator">1</option>
      <option name="trendColorInterpretation">standard</option>
      <option name="trendDisplayMode">absolute</option>
      <option name="useColors">0</option>
      <option name="useThousandSeparators">1</option>
    </single>
  </panel>
 </row>
 <row>
  <panel>
   <chart>
    <search base="s1">
     <query>eventstats count AS total_by_when BY when
      | stats count AS total BY centroide_RXB, centroide_TXB, when, total_by_when
      | eval pctg=total/total_by_when
      | table when, centroide_RXB, centroide_TXB, pctg, total</query>
    </search>
    <option name="charting.axisLabelsX.majorLabelStyle.overflowMode">ellipsisNone</option>
    <option name="charting.axisLabelsX.majorLabelStyle.rotation">0</option>
    <option name="charting.axisTitleX.visibility">visible</option>
    <option name="charting.axisTitleY.visibility">visible</option>
    <option name="charting.axisTitleY2.visibility">visible</option>
    <option name="charting.axisX.scale">linear</option>
    <option name="charting.axisY.scale">linear</option>
    <option name="charting.axisY2.enabled">0</option>
    <option name="charting.axisY2.scale">inherit</option>
    <option name="charting.chart">bubble</option>
    <option name="charting.chart.bubbleMaximumSize">50</option>
    <option name="charting.chart.bubbleMinimumSize">10</option>
    <option name="charting.chart.bubbleSizeBy">area</option>
    <option name="charting.chart.nullValueMode">gaps</option>
    <option name="charting.chart.showDataLabels">none</option>
    <option name="charting.chart.sliceCollapsingThreshold">0.01</option>
    <option name="charting.chart.stackMode">default</option>
    <option name="charting.chart.style">shiny</option>
    <option name="charting.drilldown">all</option>
    <option name="charting.layout.splitSeries">0</option>
    <option name="charting.layout.splitSeries.allowIndependentYRanges">0</option>
    <option name="charting.legend.labelStyle.overflowMode">ellipsisMiddle</option>
    <option name="charting.legend.placement">right</option>
    <option name="charting.axisX.maximumNumber">$maxRXB$</option>
    <option name="charting.axisY.maximumNumber">$maxTXB$</option>
    <drilldown>
     <set token="wRXB">$row.centroide_RXB$</set>
     <set token="wTXB">$row.centroide_TXB$</set>
     <set token="wWhen">$row.when$</set>
    </drilldown>
   </chart>
  </panel>
 </row>
 <row>
  <panel>
   <table>
    <title>Centroide escogido - $wWhen$: ($wRXB$, $wTXB$)</title>
    <search base="s1">
     <query>search centroide_RXB=$wRXB$ centroide_TXB=$wTXB$ when=$wWhen$| table SSTS, DEVICENAME, OTN, ANI, DNIS, TXB, RXB, FGS, MID0, TID, HOST_ADDR</query>
    </search>
   </table>
  </panel>
 </row>
</form>
```

Bibliografía

[i] Data, data everywhere. The Economist. Febrero 2010, http://www.economist.com/node/15557443 (Último Acceso: 2017/09/28)

[ii] DNA machine can sequence human genomes in hours. Julia Kollewe. The Guardian. Febrero 2012. https://www.theguardian.com/science/2012/feb/17/dna-machine-human-sequencing (Último Acceso: 2017/09/28)

[iii] How Many "V"s in Big Data – The Characteristics that Define Big Data. Bill Vorhies. Octubre 2013. http://data-magnum.com/how-many-vs-in-big-data-the-characteristics-that-define-big-data/ (Último Acceso: 2017/09/28)

[iv] Cloud-Based Big Data Analytics – A Survery of Current Research and Future Directions. Samiya Khan, Kashish Ara Shakil and Mansaf Alam. Department of Computer Science. Jamia Millia Islamia, New Delhi, Agosto 2015. https://arxiv.org/ftp/arxiv/papers/1508/1508.04733.pdf (Último Acceso: 2017/09/28)

[v] Parallel Processing of cluster by Map Reduce. Madhavi Vaidya. Vivekanand College. 2012. http://airccse.org/journal/ijdps/papers/0112ijdps13.pdf (Último Acceso: 2017/09/28)

[vi] MapReduce: Simplified Data Processing on Large Clusters. Jeffrey Dean, Sanjay Ghemawat, Google Inc., 2004 https://static.googleusercontent.com/media/research.google.com/es//archive/mapreduce-osdi04.pdf (Último Acceso: 2017/09/28)

[vii] Better explaining the CAP Theorem. Lior Messinger. Febrero 2014. https://dzone.com/articles/better-explaining-cap-theorem (Último Acceso: 2017/09/28)

[viii] How to beat the CAP theorem. Nathan Marz. Octubre 2011. http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html (Último Acceso: 2017/09/28)

[ix] Questioning the Lambda Architecture. Jay Kreps. Julio 2014. https://www.oreilly.com/ideas/questioning-the-lambda-architecture (Último Acceso: 2017/09/28)

[x] https://en.wikipedia.org/wiki/Syslog (Último Acceso: 2017/09/28)

[xi] Syslog Server Sizing Estimate. Cisco.com. Diciembre 2009. https://www.cisco.com/c/en/us/products/collateral/services/high-availability/white_paper_c11-557812.html (Último Acceso: 2017/09/28)

[xii] Capacity Planning Manual. Splunk Enterprise v.6.6.2. http://docs.splunk.com/Documentation/Splunk/6.6.2/Capacity/Referencehardware (Último Acceso: 2017/09/28)

[xiii] Ministerio de Empleo y Seguridad Social – Bases y tipos de cotización 2017. http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm (Último Acceso: 2017/09/28)